SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2599

NENADZIRANO UČENJE OPTIČKOG TOKA

Antonio Pavliš

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2599

NENADZIRANO UČENJE OPTIČKOG TOKA

Antonio Pavliš

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

DIPLOMSKI ZADATAK br. 2599

Pristupnik:	**Antonio Pavliš (0036497958)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Nenadzirano učenje optičkog toka**

Opis zadatka:

Procjenjivanje optičkog toka neriješen je problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolja rješenja postižu se dubokim konvolucijskim modelima. Posebno su zanimljivi nenadzirani pristupi kod kojih učenje provodimo na neoznačenim snimkama. U okviru rada, potrebno je proučiti konvolucijske arhitekture za optički tok. Oblikovati nenadzirani postupak učenja te ga primijeniti na javno dostupnim skupovima slika. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja. Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2021.

Zahvaljujem se mentoru prof. dr. sc. Siniši Šegviću i asistentu Marinu Oršiću na pomoći i stručnom vodstvu pri izradi ovog diplomskog rada.

SADRŽAJ

1. Uvod

Problem procjene optičkog toka jedan je od neriješenih problema iz područja računalnog vida. Procjenom optičkog toka želimo odrediti i ispravno prikazati kretanja objekata na sceni na temelju dvije ili više uzastopnih slika. Unatoč brojnim pokušajima rješavanja ovog problema, računalna procjena optičkog toka još uvijek nije na razini čovjeka kojemu se ovaj problem može činiti trivijalan. Čak i najbolji modeli koji pokušavaju riješiti problem procjene optičkog toka su ograničeni u procjenama i imaju problema s objektima koji se brzo kreću ili su zaklonjeni kao i sa zamućenjem uslijed kretanja (eng. motion blur) te podeksponiranim ili preeksponiranim dijelovima scene. Pokušaji rješavanja problema su brojni zbog široke primjene optičkog toka. Neke od generičkih primjena optičkog toka su u području robotike pri određivanju trenutka moguće kolizije (engl. time-to-contact), određivanja dubine scene i kretanja objekata u sceni [5] te u segmentaciji slike [1], a neke specifične primjene podrazumijevaju korištenje toka u analizi prometa i kretanju vozila ili prepoznavanju ljudskih gesti i pokreta u video sekvencama [7].

Cilj ovog rada je analizirati mogućnost provedbe nenadziranog postupka učenja na RAFT [43] arhitekturu koja, u trenutku pisanja, postiže stanje tehnike na mjerodavnim podatkovnim skupovima među ostalim nadziranim modelima. Ovaj rad će se baviti implementacijom postupka nenadziranog učenja te evaluacijom dobivenih rezultata.

Drugo poglavlje rada će definirati problem optičkog toka, a zatim ćemo napraviti pregled postojećih postupaka i modela koji pristupaju rješavanju problema optičkog toka. U trećem poglavlju ćemo opisati najvažnije dijelove dubokih konvolucijskih modela dok će četvrto poglavlje detaljnije opisati arhitekturu modela korištenog u ovom radu. Nakon toga u petom poglavlju detaljno opisujemo predloženi nenadzirani postupak učenja. Spomenut ćemo skupove podataka koji se standardno koriste u učenju i evaluaciji modela za optički tok te ćemo u sedmom poglavlju na njima evaluirati rezultate dobivene predloženim postupkom i usporediti ih s drugim poznatim modelima.

2. Optički tok

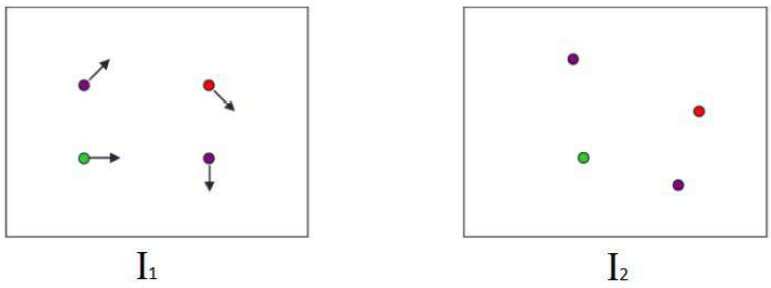
U ovom poglavlju ćemo pobliže definirati problem optičkog toka koji je predmet ovog rada. U literaturi je moguće pronaći različite definicije optičkog toka [33, 12, 6, 39], ali većina je vođena istom idejom. Optički tok je polje dvodimenzionalnih pomaka koje opisuje prividno kretanje objekata, površina i rubova u slici uzrokovano relativnim gibanjem između promatrača i scene. Navedena definicija optičkog toka počiva na pretpostavci da su intenziteti osvjetljenja piksela tijekom gibanja očuvani te da na sceni ne postoje zaklanjanja. Optički tok u modernim podatkovnim skupovima nadilazi tu definiciju budući da u njima nalazimo pojave kao što su zaklanjanja i aberacije pa time navedena definicija postaje nepotpuna. Zbog toga, preciznija definicija govori da optički tok opisuje pomake projekcija stvarnih točaka u prostoru. Dakle, procjena optičkog toka svodi se na pronalazak funkcije preslikavanja f pojedinih projekcija piksela između uzastopnih slika. Neka su $I_1, I_2 \in \mathbb{R}^{H \times W \times 3}$ dvije slike u RGB sustavu boja na temelju kojih želimo procjeniti optički tok. Funkcija preslikavanja f na ulazu prima dvije slike u navedenom formatu, a na svom izlazu generira dvodimenzionalno vektorsko polje koje predstavlja pomake projekcija svakog pojedinog piksela.

$$f : \{I_1 \times I_2 \mid I_1, I_2 \in \mathbb{R}^{H \times W \times 3}\} \rightarrow \mathbb{R}^{H \times W \times 2} \tag{2.1}$$

Rezultantno vektorsko polje je definirano u svakom pikselu izvorne slike. Za takvo polje kažemo da je gusto polje, a pripadni optički tok je gusti optički tok (engl. dense optical flow). Osim toga poznajemo i rijetki optički tok (engl. sparse optical flow) gdje vektor pomaka nije definiran u svim pikselima nego u određenom podskupu piksela. Strujnice vektorskog polja pokazuju smjer relativnog gibanja piksela što je prikazano na slici 2.1. Ako strujnica polja na mjestu (x, y) ima oblik $f(I_1, I_2)_{xy} = (\Delta x, \Delta y)$, tada vrijedi:

$$I_1(x, y) \approx I_2(x + \Delta x, y + \Delta y) \tag{2.2}$$

Navedena jednadžba povezuje pripadajuće piksele dvije slike koji predstavljaju



Slika 2.1: Prikaz relativnih pomaka piksela i pripadajućih vektora optičkog toka opisanih jednadžbom 2.2. Slika je preuzeta iz [2].

projekciju iste točke objekta, pomaknutog u odnosu na promatrača. Bitno je primjetiti da u jednadžbi 2.2 ne vrijedi stroga jednakost. Razlog tomu su zamućenja, aberacije i promjene osvjetljenja koja se pojavljuju u modernim podatkovnim skupovima, a upravo zbog toga rekonstrukcijski postupci nemaju šansu postići visoki rezultat u procjeni optičkog toka.

2.1. Izvod jednadžbe optičkog toka

Ovdje ćemo prikazati izvod jednadžbe optičkog toka (engl. Optical flow constraint equation) temeljen na pretpostavci konstantnog intenziteta piksela tijekom relativnog pomaka.

Označimo sada s $I(x, y, t)$ funkciju koja određuje intenzitet piksela na poziciji (x, y) u trenutku t. Prema pretpostavci da intenzitet osvjetljenja piksela nakon pomaka ostaje nepromijenjen, piksel (x, y) nakon pomaka za $(\Delta x, \Delta y)$ zadovoljava jednadžbu:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.3}$$

Razvojem funkcije s desne strane jednakosti iz jednadžbe 2.3 u Taylorov red, za male pomake vrijedi jednakost:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \cong I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \epsilon \tag{2.4}$$

U 2.4, ϵ predstavlja ostatak koji je jednak sumi članova drugog i svih viših redova Taylorovog reda koje možemo zanemariti budući da smo pretpostavili proizvoljno male pomake piksela.

Uvrštavanjem 2.4 u 2.3 dobivamo izraz:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{2.5}$$

odnosno

$$\frac{\partial I}{\partial t} = \frac{-(\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y)}{\Delta t} \tag{2.6}$$

Uvedimo oznake:

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}, u_x = \frac{\Delta x}{\Delta t}, u_y = \frac{\Delta y}{\Delta t}$$

Sada izraz 2.6 izgleda:

$$I_t = -(I_x u_x + I_y u_y) \tag{2.7}$$

Uz limes $\lim_{\Delta t \to 0} u_x$, odnosno $\lim_{\Delta t \to 0} u_y$, izrazi u_x i u_y predstavljaju brzinu promjene x i y koordinata piksela u vremenu.

$$\mathbf{u} = (u_x, u_y) = (\dot{x}, \dot{y}) \tag{2.8}$$

U matričnom obliku, jednadžbu optičkog toka možemo zapisati:

$$\nabla I \cdot \mathbf{u} + \frac{\partial}{\partial t} I = 0 \tag{2.9}$$

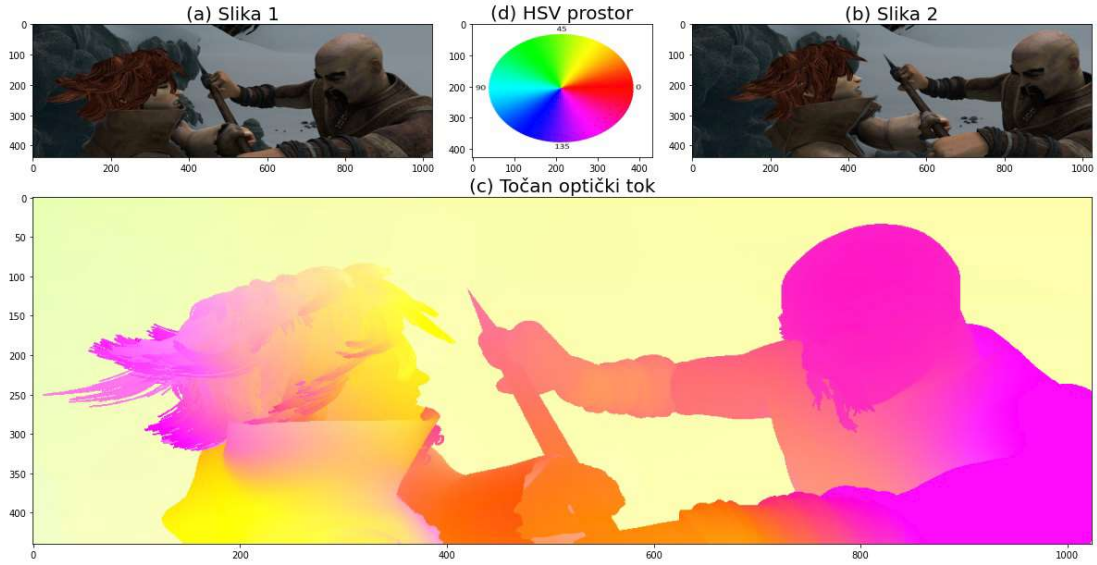
Izvod jednadžbe preuzet iz [39, 12, 4].

2.2. Vizualizacija optičkog toka

Kako bismo vizualizirali vektore optičkog toka, koristimo HSV prostor boja. HSV prostor boja se sastoji od 3 komponente: ton boje, zasićenje boje i svjetlina. Ton ili nijansa boja je predstavljena kutom od 0 do 360 stupnjeva. Zasićenje boje predstavlja količinu sive boje, a kreće se u intervalu od 0 do 1. Izbijeljene boje dobivamo smanjenjem zasićenja prema nuli dok se primarna boja postiže zasićenjem jednakim 1. Vrijednost ili svjetlina zajedno sa zasićenjem opisuje svjetlinu ili intenzitet boje od 0 do 1, gdje je 0 potpuno crna, a 1 otkriva najveću svjetlinu boje.

Optički tok možemo vizualizirati preslikavanjem vektora pomaka pojedinih piksela u HSV prostor boja. Kut između vektora pomaka i osi x u dvodimenzionalnom koordinatnom sustavu preslikavamo u prvu komponentu HSV sustava boja te izravno određujemo nijansu boje kojom će vektor biti prikazan. Normu vektora preslikavamo

u interval $[0, 1]$ te tako određujemo zasićenost i svjetlinu boje u prikazu. Rezultat opisanog preslikavanja je prikazan na slici 2.2.



Slika 2.2: HSV sustav boja (d) i vizualizacija vektora pomaka pojedinih piksela (c) preslikavanjem kuta zakreta vektora i norme u HSV sustav boja iz (d). Ulazni par slika (a) i (b) je iz skupa Sintel train.

2.3. Pregled modela za optički tok

Zadatak procjene optičkog toka je problem u području računalnog vida koji još uvijek nije u potpunosti riješen. Autori su tijekom godina nudili različita rješenja, više ili manje uspješna. Suština problema je pronaći funkciju preslikavanja pojedinih piksela s obzirom na njihova relativna kretanja unutar slike [6]. Takva funkcija treba generirati preslikavanja u obliku vektorskog polja. Uspješnost procjene evaluiramo usporedbom s točnim (engl. ground truth) vektorskim poljem te ukoliko problem formuliramo kao zadatak minimizacije razlike procijenjenih i točnih tokova preko cijelog skupa za učenje, dobivamo klasični optimizacijski problem s ciljem minimizacije funkcije gubitka:

$$L(\mathbf{f}) = ||\mathbf{f}_{gt} - \mathbf{f}|| \tag{2.10}$$

Prvi pokušaji rješavanja problema optičkog toka su se temeljili upravo na pronalaženju ovakvog postupka optimizacije. Horn i Schunck [16] predlažu postupak optimizacije koji se temelji na uparivanju piksela konzistentnih intenziteta u parovima slika. Postupak je osjetljiv na promjene u intenzitetu piksela koje se događaju uslijed zaklanjanja, promijene osvjetljenja odnosno sjena kao i na velike pomake objekata. Brox

i Malik [8] unaprijeđuju ovakav postupak uparivanjem značajki umjesto izvornih piksela te postižu bolju procjenu toka u pikselima sa velikim pomakom. Velika mana ovih postupaka je činjenica da je optimizacijski postupak u potpunosti ručno (engl. handcrafted) definiran i kao takav može imati velika ograničenja u mogućnosti pronalaženja optimuma. Još jedan primjer takvog postupka je Full Flow [9]. Full Flow provodi diskretan optimizacijski postupak nad cijelim poljem pomaka uz linearnu asimptotsku složenost, a bitna značajka je rad postupka bez potrebe za uparivanjem deskriptora pojedinih piksela.

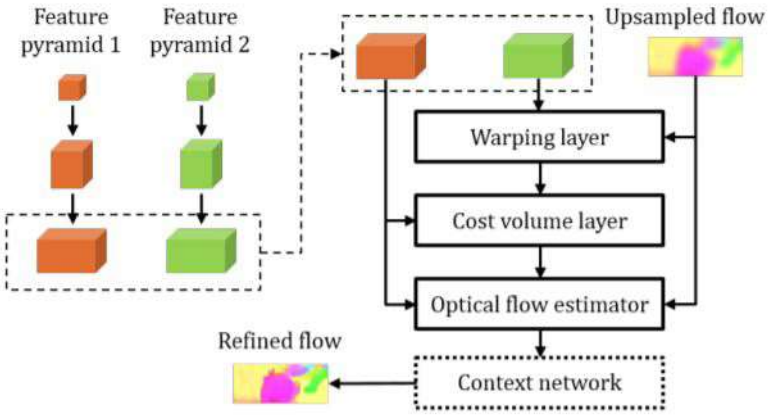
Općenito, problem optimizacije procjene toka povlači problem postizanja kompromisa između uparivanja piksela jednakog intenziteta, tj. vizualne konzistentnosti (engl. data term) i regularizacije koja osigurava prostornu glatkoću procjenjenog toka (engl. regularization term) [40]. Postupak TV-L1 [34] koristi L1 normu prilikom računanja vizualne sličnosti za razliku od prethodnih navedenih radova koji su koristili kvadrat pogreške. Ukupna varijacija (engl. total variation - TV) je mjera složenosti slike koju koristimo kao regularizacijski član tijekom postupka učenja. Regularizacijom nastojimo postići glatku rezultantnu funkciju bez naglih skokova koji su uzrokovani šumom.

Rastom popularnosti neuronski mreža, posebice dubokih modela, pojavili su se pokušaji rješavanja ovog problema dubokim konvolucijskim mrežama [2] koje su kasnije pokazale najveću uspješnost. Zadatak pronalaska funkcije preslikavanja piksela povjeren je internim slojevima duboke neuronske mreže koji na temelju izlučenih karakterističnih značajki implicitno uparuju pripadajuće piksele iz parova slika. Kažemo da takve modele učimo s kraja na kraj (engl. end-to-end). Jedan od prvih pokušaja korištenja neuronske mreže u tu svrhu predlažu Zhou et al. (1988.) u [46].

Česta podatkovna reprezentacija kojom se služe modeli za optički tok je volumen cijene (engl. cost volume). Volumen cijene je tenzor četvrtog reda koji sadrži sličnosti svih parova piksela iz dvije uzastopne slike. Procjena optičkog toka iz volumena cijene je diskretni optimizacijski problem. Problem ovakvog pristupa može biti ogroman prostor pretraživanja budući da piksel može biti uparen sa bilo kojim iz druge slike [32, 9].

Veliki broj radova uvodi neku vrstu iterativnog ponašanja tijekom procjene toka [18, 35, 41, 43, 19, 26]. Osim toga, neki autori predlažu korištenje više od dvije uzastopne slike u procjeni toka [28, 21].

U nastavku ćemo pobliže opisati svojstva i specifičnosti dubokih modela za optički tok podijeljenih po paradigmi učenja.



Slika 2.3: Blok shema jednog sloja piramidalne arhitekture PWC-Net modela. Osnovni elementi jedne razine piramide su sloj izobličavanja slike na temelju preuzorkovanog (engl. upsampled) toka iz prethodne razine piramide, korelacijski sloj koji pohranjuje sličnosti uparenih piksela, sloj za procjenu toka na trenutnoj razini te sloj za izlučivanje konteksta. Svaka razina piramide koristi tok dobiven u prethodnoj razini te ga unaprijeđuje na višoj rezoluciji. Slika je preuzeta iz [41].

2.3.1. Nadzirani modeli

Osnovni zahtjev nadziranih modela za optički tok je postojanje označenih (engl. ground truth) podataka u skupu za učenje što ponekad nije praktično jednostavno. FlowNet [11] i FlowNet 2.0 [20] predstavljaju najranije primjere korištenja dubokih konvolucijskih modela za potrebe procjene optičkog toka. Novost koju su uveli ovi modeli je korištenje koder-dekoder (engl. encoder-decoder) arhitekture konvolucijske mreže. Neka od poboljšanja su dobivena korištenjem piramidalnog pristupa obradi od grubog do finog (engl. coarse-to-fine) [35, 18, 45]. PWC-Net [41] je ostavio veliki trag među konvolucijskim modelima za optički tok. Uz korištenje svih prije navedenih principa, postigao je stanje tehnike korištenjem manjeg broja parametara u modelu. Upravo zbog toga su mnogi kasniji radovi koristili dijelove PWC-Net arhitekture kao što su primjerice, nama zanimljivi, Selflow [28] i UFlow [22]. Jedan sloj PWC-Net modela je prikazan na slici 2.3.

LiteFlowNet [18] unaprijeđuje FlowNet 2 [20] smanjenjem veličine modela 30 puta uz postignute bolje rezultate na mjerodavnim skupovima. Karakteristike LiteFlowNeta su korištenje prije spomenute piramidalne obrade, ali i tehnike izobličavanja (engl. warping) kojom svaki piksel pomičemo u smjeru trenutne procjene toka te tako dobivenu sliku uspoređujemo s izvornom u svrhu evaluacije kvalitete procijenjenog

toka. Prostorno transformiranje piksela ili značajki je poznata tehnika u mnogim radovima [41, 35, 20].

Hur et al. [19] predlažu iterativnu tehniku temeljenu na dijeljenju parametara koja može biti kombinirana s različitim arhitekturama mreža. Tehnika, između ostalog, uključuje procjenu toka u oba smjera (naprijed-natrag) i pomaže poboljšanju procjene toka uz smanjenje broja parametara. Procjena toka u oba smjera je tehnika koju su prihvatili mnogi kasniji radovi [31, 27].

Model korišten u ovom radu je RAFT: Recurrent All-Pairs Field Transforms [43]. RAFT je iterativan model koji kombinira dobro provjerene tehnike kao što su formiranje volumena cijene sa sličnostima među parovima piksela i iterativan pristup. Budući da je iterativan, konačan tok dobiva postupnim ugađanjem primjenom povratnog modula. U trenutku objave rada, RAFT postiže stanje tehnike među nadziranim modelima na mjerodavnim podatkovnim skupovima. Detaljnu arhitekturu modela RAFT opisujemo u sljedećem poglavlju.

2.3.2. Nenadzirani modeli

Nedostatak točnih podataka za učenje potaknuo je razvoj nenadziranih modela za optički tok. U suštini svakog nenadziranog postupka, optimizacijski zadatak je minimizirati grešku koja se pojavljuje zbog razlike između izvorne i transformirane slike. Nenadzirani modeli temelje se na dobro poznatim arhitekturama mreža u pozadini uz izmjene u paradigmi učenja. Trenutno jedan od najefikasnijih nenadziranih modela, SelFlow [28], koristi PWC-Net [41] arhitekturu mreže kao i UFLow [22].

Jonschkowski et al. analiziraju utjecaj temeljnih elemenata nenadziranih modela [22], a to su fotometrijski gubitak, način upravljanja zaklanjanjima, regularizacija [23] i samonadzor. Pristupi formiranju nenadziranih postupaka se mnogo razlikuju od autora do autora, primjerice OAFlow [44] uključuje regularizacijski gubitak dok DDFlow [27] regularizira postupak učenja kroz vlastiti samonadzor. UnFlow [31] provodi procjenu toka u oba smjera (naprijed-natrag) te na temelju provjere konzistentnosti unaprijednog i povratnog toka određuje zaklonjena područja koja isključuje tijekom izračuna funkcije gubitka. Funkcija gubitka je fotometrijski gubitak nad slikama koje su obrađene Census transformacijom [14]. Census transformacija je široko prihvaćen postupak među nenadziranim postupcima učenja optičkog toka. Census transformacija u izvornom obliku nije diferencijabilna pa se u implementacijama koriste njezine aproksimacije izvedene pomoću konvolucijskih filtara odgovarajućih veličina. Gotovo svi autori predlažu korištenje Census transformacije u svojim fotometrijskim gubicima

[28, 27, 31, 44] dok Jonschkowski et al. analiziraju utjecaje drugih mogućnosti kao što su SSIM, Charbonnier i L1 gubici [22].

Treba primjetiti da UnFlow nema mehanizam simuliranja zaklanjanja i samonadzora za razliku od DDFlowa-a [27] i SelFlow-a [28] koji to ostvaruju dodavanjem umjetnih zaklanjanja i samonadzora pri učenju toka u tim područjima. Postupci koji uključuju samonadzor u zaklonjenim područjima [27, 28, 22] spadaju u postupke samonadziranog učenja (engl. self-supervised learning) koje je podskup područja nenadziranog učenja. Lifshitz et al. (2020.) predstavljaju metodu tzv. odmotavanja funkcije gubitka (engl. cost function unrolling) koja nediferencijabilna ograničenja u obliku L1 norme transformira u diferencijabilnu funkciju gubitka [26]. Autori pokazuju da metoda pomaže u preciznosti procjene toka.

Janai et al. [21] koriste tri uzastopne slike u nenadziranom postupku učenja optičkog toka. Korištenje više od dvije uzastopne slike donosi više informacija o kretanjima u sceni i zbog toga je ta tehnika prihvaćena u nekim radovima [28].

3. Glavni dijelovi dubokih modela

Duboko učenje je dio strojnog učenja orjentiran na specifičnosti velikih skupova podataka te podataka koji su strukturno složeniji kao što su slike, govorni signali ili pisani tekst. Duboki modeli se sastoje od slojeva koji su u suštini parametarske funkcije koje se primjenjuju na ulazni podatak te se izlaz funkcije prosljeđuje kao izlaz pojedinog sloja. Uzastopnim nizom različitih slojeva postižemo transformacije ulaznih podataka koje mogu biti linearne ili nelinearne, ovisno o izboru funkcija, odnosno slojeva. Korištenjem većeg broja slojeva, rješavanje željenog problema razlažemo na manje cjeline te je svaki sloj zadužen za pojedini dio problema. Treniranjem modela postavljamo parametre funkcija, odnosno slojeva, na optimalne vrijednosti kako bi niz transformacija koje djeluju na ulazni podatak generirao željeni izlaz. U tom procesu nam pomažu različiti oblici gradijentne optimizacije. U nastavku ćemo opisati vrste slojeva koji se tipično pojavljuju u dubokim modelima.

Potpuno povezani sloj

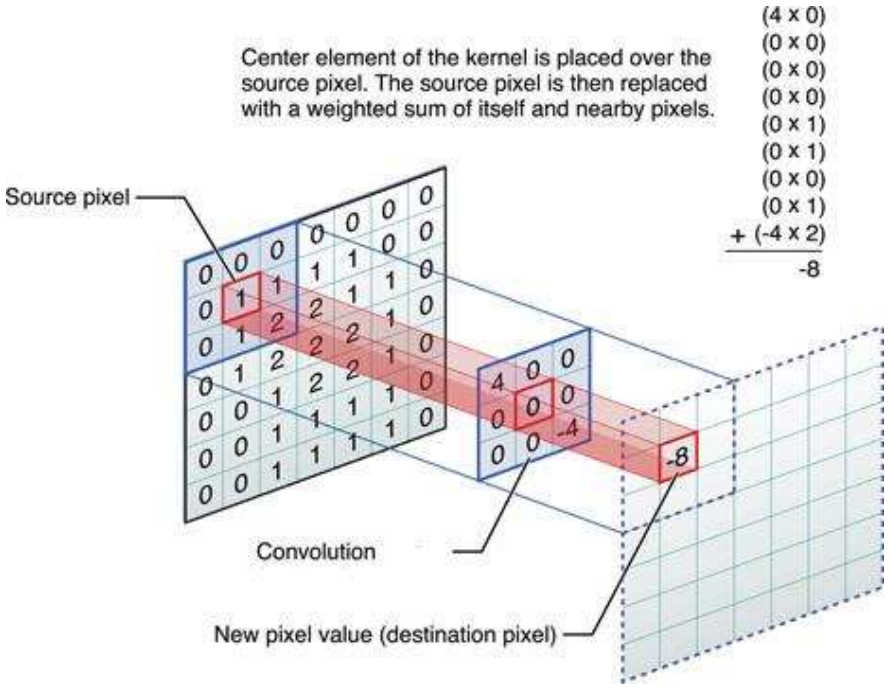
Ova vrsta slojeva se najčešće pojavljuje u neuronskim mrežama. Funkcija koju obavlja potpuno povezani sloj može se opisati afinom transformacijom ulaznog podatka x. Svaki izlaz sloja se prosljeđuje na svaki ulaz sljedećeg sloja uz pripadajuću težinu. Funkciju koju ovaj sloj obavlja možemo zapisati kao:

$$f(x; W, b) = \sigma(Wx + b) \tag{3.1}$$

U prikazanoj jednadžbi x označava ulazni podatak, W pripadajuće težine, b pripadajući pomak, a σ aktivacijsku funkciju koja se primjenjuje na rezultat afine transformacije.

Konvolucijski sloj

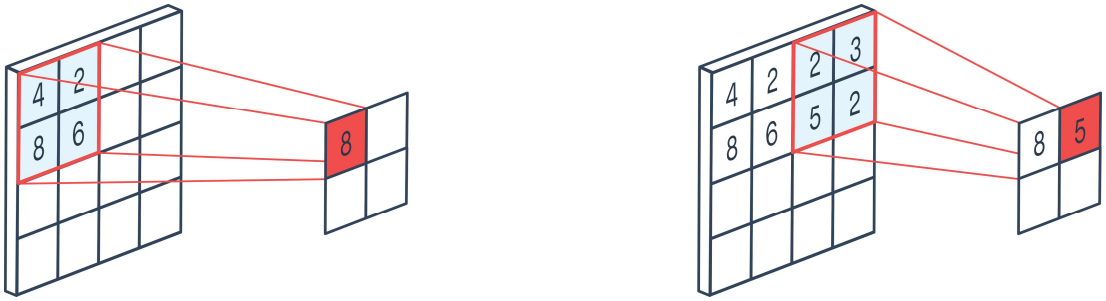
Najvažniji dio dubokih modela je upravo konvolucijski sloj. Ideja ovog sloja je pronalaženje nekih specifičnih uzoraka u ulaznim podacima. Funkciju koju obavlja konvolucijski sloj možemo opisati matematičkom operacijom unakrsne korelacije definirane kao:



Slika 3.1: Primjena jezgre konvolucijskog sloja na dio ulazne slike. **Izvor:** https://miro.medium.com/max/464/0*e-SMFTzO8r7skkpc

$$h(t) = (W * x)(t) = \int_{D(W)} W(\tau)X(t+\tau)d\tau \tag{3.2}$$

Iz jednadžbe 3.2 je vidljivo da se izlaz sloja dobiva kao skalarni produkt dijela ulaza x i jezgre W koja prekriva određeni dio ulaza, a za svaki sljedeći izlazni element jezgru pomičemo po prostornim dimenzijama ulaza za definirani broj koraka. Zbog toga, elemente jezgre možemo promatrati kao težine sloja koje množe elemente ulaznog podatka. Princip je dakle vrlo sličan potpuno povezanom sloju s tom razlikom da jezgra ne prekriva cijeli ulazni podatak već samo njegov dio te tako u obzir uzimamo lokalna susjedstva piksela i specifičnosti koje donose međudjelovanja susjednih piksela. Osim toga, razlika u odnosu na potpuno povezani sloj je dijeljenje težina što znači da se iste težine jezgre koriste nad svim lokalnim područjima. Obično se rezultatu konvolucije dodaje pomak (engl. bias) koji je također zajednički svim lokalnim područjima. Očito je da jedna operacija konvolucije zahtjeva puno manje parametara od potpuno povezanog sloja budući da su težine jezgre zajedničke nad svim lokalnim područjima za razliku od potpuno povezanog sloja gdje je svakom ulaznom elementu pridružena odgovarajuća težina. Izlaz konvolucijskog sloja nazivamo mapom značajki, a jedan sloj tipično provodi konvoluciju nad ulazom primjenom različitih jezgara te tako dobivamo niz mapa značajki, po jednu za svaku korištenu jezgru.



Slika 3.2: Sažimanje maksimumom (engl. max pooling) u oknu veličine 2×2 s korakom 2. **Izvor:** https://peltarion.com/static/2d_max_pooling_pa1.png

Operacija konvolucije može biti definirana za različite dimenzionalnosti ulaznih podataka. Tako se u slučaju slikovnih podataka provodi dvodimenzionalna konvolucija dijela slike i dvodimenzionalne jezgre čije su veličine u dubokim modelima tipično 3×3, 7×7 i slične. Slika 3.1 prikazuje korak konvolucije ulazne slike sa jezgrom dimenzija 3×3. Treba primjetiti da se primjenom konvolucije na ulazni podatak, veličina izlaznog podatka smanjuje ovisno o veličini jezgre i koraku konvolucije. U svrhu očuvanja prostorne veličine podataka, tipično se izlazni podatak nadopunjuje po rubovima (engl. padding).

Sloj sažimanja

Za reduciranje prostorne veličine podataka u dubokim modelima koriste se slojevi sažimanja (engl. pooling). Sažimanje se provodi oknom jezgre koja se određenim korakom pomiče po prostornim dimenzijama ulaznog podatka i na svom izlazu generira rezultat koji je dobiven primjenom određene funkcije. Taj rezultat je tipično skalar te se cijelo ulazno područje obuhvaćemo jezgrom sažima u taj jedan skalar. Funkcije koje se provode su tipično funkcija prosjeka (engl. average pooling), maksimuma (engl. max pooling) i slične. U slučaju rada s podacima kao što su slike, jezgra sažimanja je dvodimenzionalna i tipično je dimenzija 2×2. Takav slučaj prikazujemo na slici 3.2

Aktivacijska funkcija

Izlaz sloja možemo shvatiti kao linearnu kombinaciju elemenata ulaza i vrijednosti pripadajućih težina. Nizanjem uzastopnih slojeva postižemo linearnu kombinciju pojedinih linearnih kombinacija što je u suštini opet linearna kombinacija. Iz tog razloga, potrebno je u model dodati nelinearnosti koje će nam omogućiti formiranje složenijih decizijskih granica na izlazu modela koje nisu linearne te time omogućiti bolju generalizaciju modela. Nelinearnosti postižemo aktivacijskim funkcijama koje primjenjujemo neposredno na izlaze pojedinih slojeva. Odabrana funkcija treba zadovoljiti sljedeća svojstva: nelinearnost, diferencijabilnost na cijelom području domene, mono-

tonost i ponašanje blisko identiteti u okolini ishodišta. Najčešće aktivacijske funkcije u dubokim modelima su sigmoida, zglobnica i tangens hiperbolni [38].

– **Sigmoida**

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

– **Zglobnica** (engl. ReLU)

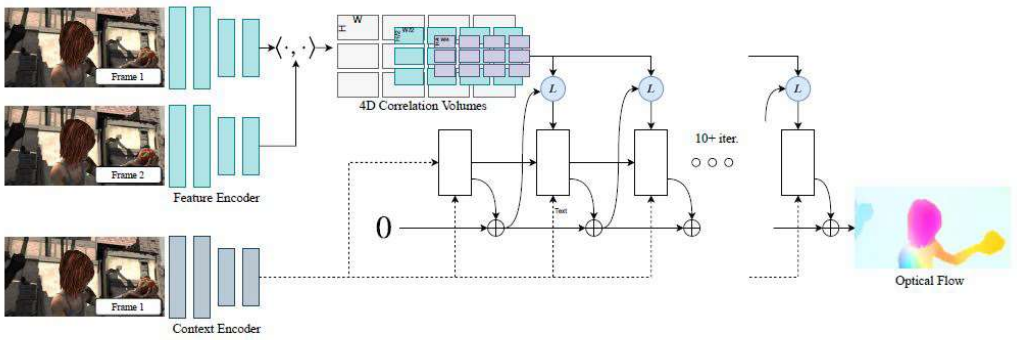
$$f(x) = \max(0, x) \tag{3.4}$$

– **Tangens hiperbolni**

$$f(x) = \tanh(x) \tag{3.5}$$

4. Procjena optičkog toka arhitekturom RAFT

Kao što je ranije spomenuto, u današnje vrijeme najbolje rezultate u rješavanju problema iz područja računalnog vida postižu duboki konvolucijski modeli. Tipični zadaci koji se pojavljuju u tom području su klasifikacija slika, prepoznavanje objekata na slikama ili video isječcima, segmentacija, procjena optičkog toka na nizovima slika i slično. U ovom poglavlju ćemo detaljnije opisati arhitekturu RAFT [43]. Radi se o dubokom konvolucijskom modelu za optički tok koji se temelji na iterativnom ugađanju procjene toka. Model su predstavili Zachary Teed i Jia Deng sa sveučilišta Princeton u svom članku iz 2020. godine. U vrijeme objave članka, model postiže stanje tehnike u procjeni optičkog toka na podatkovnim skupovima KITTI i Sintel. Osim visoke preciznosti u procjeni toka, model odlikuje iznimna sposobnost generalizacije na neviđenim podacima te efikasnost u iskorištavanju računalnih resursa kao i mali broj iteracija potrebnih tijekom faze treniranja budući da model relativno brzo konvergira. Za usporedbu s ostalim modelima za optički tok, faza treniranja RAFT-a zahtjeva deset puta manje iteracija. U načelu, RAFT se sastoji od tri glavne komponente: (1) koder značajki, (2) korelacijski sloj i (3) modul za ažuriranje. Komponente su međusobno nezavisne u implementaciji, a njihovim međudjelovanjem u unaprijednom prolazu dobijemo model koji tok procjenjuje na visokoj rezuloci ji što ga razlikuje od ostalih modela koji iterativno procjenjuju od grube prema finijim rezolucijama (engl. coarse-to-fine). Njihova ideja je ugrubo procjeniti tok na vrlo niskoj rezoluciji te ga skalirati na originalnu rezoluciju i ugoditi, a tipičan primjer takvog modela je PWC-Net. Nedostaci takvog pristupa su nemogućnost oporavka od pogreške u procjeni toka koja se dogodila na niskoj rezoluciji, velika vjerojatnost propusta malih objekata koji se brzo kreću u sceni te veliki broj potrebnih iteracija u fazi treniranja takvih modela. RAFT je dizajniran s ciljem rješavanja ovih problema. Prva komponenta RAFT-a, koder značajki, izlučuje značajke pojedinih piksela ulaznih slika. Korelacijski sloj služi za izračun vizualnih sličnosti među svim parovima piksela ulaznih slika, a modul za



Slika 4.1: Blok shema RAFT. Prvi element u cjevovodu modela je koder značajki čija je zadaća izlučivanje značajki ulazne slike u vektor značajki. Posebno izlučene značajke slike 1 nazivamo kontekst koji se koristi u svakoj iteraciji algoritma. Korištenjem dobivenih vektora značajki konstruiramo korelacijsko polje tj. volumen cijene na način da množimo sve parove vektora značajki na različitim rezolucijama. Ugađanje toka se provodi uzastopnim iteracijama. Pretraživanje korelacijske piramide obavlja operator L koji indeksira iz korelacijske piramide na temelju trenutne procjene toka. Slika je preuzeta iz [43].

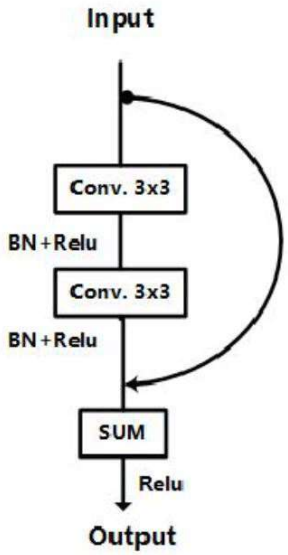
ažuriranje iterativno ugađa tok u unaprijed određenom broju ponavljanja. Blok shema RAFT-a je prikazana na slici 4.1. Važno svojstvo operatora ažuriranja su dijeljene težine tijekom iteracija što ovaj model razlikuje od ostalih modela koji uključuju neku vrstu iterativnog ponašanja i ograničeni su u broju iteracija ili je broj iteracija fiksno postavljen. RAFT može provoditi do 100 iteracija ugađanja bez divergencije.

4.1. Komponente arhitekture RAFT

Prije opisa metode rada RAFT modela, opisati ćemo slojeve modela koju su autori predložili. Arhitektura se dijeli na 3 logičke cjeline: koder značajki, korelacijski sloj te modul za ažuriranje.

4.1.1. Koder značajki

Zadaća kodera značajki je izlučivanje vizualnih značajki iz para slika dovedenih na ulaz. Dobivene značajke koristimo za izračun međusobne sličnosti piksela prve slike sa svim pikselima iz druge slike. Na temelju dobivenih sličnosti kasnije uparujemo najsličnije parove i procjenjujemo kretanje piksela u sceni, odnosno procjenjujemo optički tok. Koder značajki je izveden kao konvolucijska mreža koja se sastoji od 6 rezidualnih blokova. Strukturu rezidualnog bloka prikazujemo na slici 4.2.

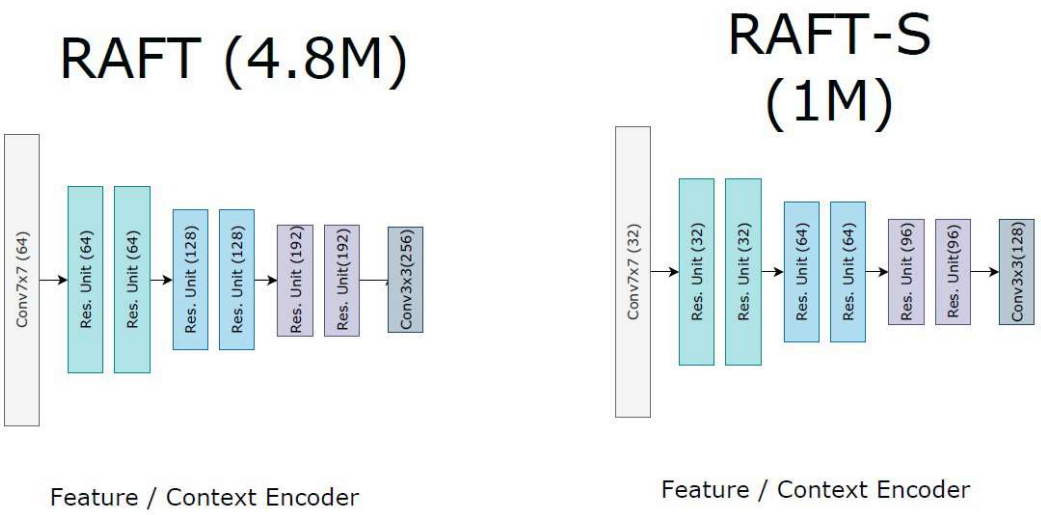


Slika 4.2: Struktura rezidualne jedinice koji je sastavni dio kodera značajki. Jedinica se sastoji od dva konvolucijska sloja sa jezgrama veličine 3×3 na čijim se izlazima nalazi normalizacija po grupi te ReLU aktivacija, a izlaz je zbroj drugog konvolucijskog sloja i nepromjenjenog ulaza u jedinicu. **Izvor:** https://www.researchgate.net/profile/Xiaoguang-Niu/publication/320631725/figure/fig2/AS:566728984678400@1512130127475/Basic-residual-unit-a-and-deformed-residual-unit-b.png

Ideja korištenja rezidualnih jedinica je stabilizacija postupka učenja. Autori su eksperimentalno pokazali da uvođenje direktnih veza među nesusjednim slojevima doprinosi stabilizaciji vrijednosti gradijenata i konvergenciji učenja, a istovremeno dobivamo glatku krivulju učenja. [15, 25].

Još jedna tehnika koja pomaže kod zadržavanja vrijednosti gradijenata u optimalnom rasponu tijekom unatražnog prolaza je normalizacija slojeva (engl. layer normalization). RAFT koristi normalizaciju samo u rezidualnim jedinicama, nakon svakog konvolucijskog sloja u koderu značajki. Normalizacija vrijednosti aktivacija pojedinih slojeva pomaže kod ubrzavanja konvergencije tijekom faze učenja budući da normalizacijom zadržavamo aktivacije slojeva oko vrijednosti nula, a upravo u tom području je većina aktivacijskih funkcija najosjetljivija te su vrijednosti gradijenata koje tada dobivamo najkorisnije. Postoje razne izvedbe normalizacije slojeva, a neke najpopularnije su: batch, group i weight normalizacije [3, 36].

Autori RAFT-a u svom radu [43] predlažu dvije implementacije [42]. Razlika je u broju parametara modela. Puni RAFT se sastoji od 4.8 miliona parametara, dok reducirana verzija sadrži 1 milion parametara, a logika modela je ista. Manju verziju RAFT-a ćemo u nastavku rada imenovati s RAFT-S (engl. RAFT small). Autori su reducirali



Slika 4.3: Arhitektura kodera značajki u dvije izvedbe RAFT modela. Obje implementacije sadrže 6 rezidualnih blokova gdje prva dva rade s $1/2$ rezolucije izvorne slike, sljedeća dva s $1/4$ rezolucije izvorne slike te posljednja dva s $1/8$ rezolucije. Ova arhitektura se ujedno koristi za značajke konteksta koje se posebno izlučuju iz prve od dvaju uzastopnih slika na ulazu [43].

neke slojeve u modelu te tako smanjili broj parametara dok je učinkovitost modela još uvijek ostala na visokoj razini. Slika 4.3 prikazuje strukturu kodera značajki u obje izvedbe modela.

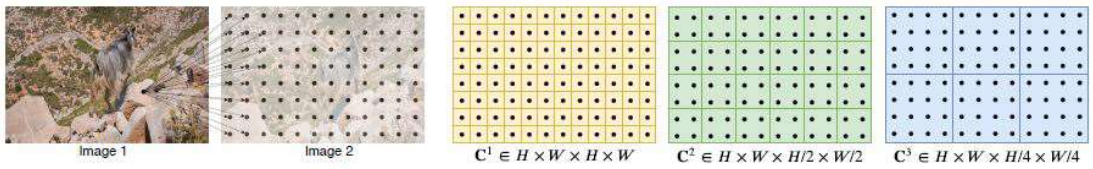
Kao što je objašnjeno u opisu slike 4.3, na izlazu kodera dobivamo izlučene značajke na rezoluciji koja je jednaka $1/8$ izvorne rezolucije slike. Dakle, funkcija g_0 koju koder izvodi ima potpis:

$$g_0 : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H/8 \times W/8 \times D} \tag{4.1}$$

U slučaju RAFT modela, D ima vrijednost 256 što znači da koder generira 256 različitih mapa značajki takvih dimenzija dok je u slučaju RAFT-S $D = 128$ pa model radi sa manje mapa značajki.

4.1.2. Korelacijski sloj

Korelacijski sloj služi za kreiranje volumena cijene koji sadrži sličnosti među svim parovima slikovnih elemenata iz uzastopnih slika. Volumen cijene kreiramo korištenjem izlučenih mapa značajki dobivenih od kodera značajki. Budući da znamo da izlučene mape značajki imaju oblik $g_0(I_1) \in \mathbb{R}^{H \times W \times D}$ i $g_0(I_2) \in \mathbb{R}^{H \times W \times D}$, volumen cijene



Slika 4.4: Konstruiranje 4D volumena cijene kao skalarni produkt značajki među svim parovima piksela. Pojedini volumen cijene ima dimenzije $H \times W \times H/2^k \times W/2^k$. Razine piramide formiramo sažimanjem srednjom vrijednosti po zadnje dvije dimenzije izvornog volumena cijene uz različite veličine jezgara. Slika prikazuje 2D odreske pojedinih razina dobivene korelacijske piramide $\{C^1, C^2, C^3, ...\}$ [43].

možemo računati kao skalarni produkt značajki po dimenziji D u svim mogućim parovima piksela. Dakle, značajke $g_0(I_1)_{ij}$ na pozicijama $(i, j, h) \in \mathfrak{D}(g_0(I_1)) \mid \forall h \in D$ skalarno množimo sa značajkama $g_0(I_2)_{kl}$ na pozicijama $(k, l, h) \in \mathfrak{D}(g_0(I_2)) \mid \forall h \in D$ na način:

$$C_{ijkl} = \sum_h g_0(I_1)_{ijh} \cdot g_0(I_2)_{klh} \mid \forall (i, j) \in \mathfrak{D}(g_0(I_1)), \forall (k, l) \in \mathfrak{D}(g_0(I_2)) \tag{4.2}$$

Jednadžba 4.2 u matričnom zapisu ima oblik:

$$C = g_0(I_1) \cdot g_0^T(I_2) \in \mathbb{R}^{H \times W \times H \times W} \tag{4.3}$$

Autori predlažu kombiniranje dobivenog volumena cijene na različitim razinama razlučivosti. To postižemo formiranjem korelacijske piramide koja će na svakoj svojoj razini sadržavati volumen cijene različite rezolucije. Svaku od tih razina dobivamo sažimanjem zadnje dvije dimenzije originalnog volumena cijene uz posebnu veličinu okna. Provodimo sažimanje prosječnom vrijednosti (engl. average pooling) uz veličinu okna jezgre u iznosima 1, 2, 4 ili 8 ovisno o razini korelacijske piramide. Postupak je prikazan na slici 4.4. Uslijed sažimanja posljednje dvije dimenzije originalnog volumena cijene, zaključujemo da volumen C^k na razini k piramide ima oblik: $C^k \in H \times W \times H/2^k \times W/2^k$. Konačno, korelacijska piramida se tada sastoji od niza: $\{C^1, C^2, C^3, C^4\}$

Kombiniranje volumena cijena na različitim rezolucijama omogućuje istovremeno praćenje informacija o kretanjima velikih i malih objekata, a zadržavanje prve dvije dimenzije volumena cijene pomaže određivanju toka na visokoj rezoluciji u svim razinama piramide unoseći informacije koje možda ne bi bile vidljive na nižim rezolucijama.

4.1.3. Operator ažuriranja

Operator ažuriranja je izveden neuronskom mrežom čiji je glavni element povratna ćelija GRU. U svakoj iteraciji ugađanja toka, provodi se jedan unaprijedni prolaz kroz navedeni operator ažuriranja koji na trenutnu procjenu $\mathbf{f_k}$ toka dodaje trenutno izračunati pomak $\mathbf{\Delta f}$ prema točki konvergencije $\mathbf{f}^*$. Odnosno, za uzastopne procjene toka vrijedi:

$$\mathbf{f_0} = \mathbf{0}$$
$$\mathbf{f_{k+1}} = \mathbf{f_k} + \mathbf{\Delta f}$$
$$\lim_{k \to \infty} \mathbf{f_k} = \mathbf{f}^*$$

GRU (eng. Gated Recurrent Unit) je vrsta povratnih neuronskih mreža koja je dizajnirana za rješavanje problema nestajućeg gradijenta (engl. vanishing gradient) koji se javlja u standardnim ponavljajućim mrežama. Arhitekturu GRU možemo vidjeti na slici 4.5.

Vrata za ažuriranje i resetiranje GRU-a ostvarujemo odgovarajućim slojevima unutar GRU ćelije. Autori RAFT-a predlažu zamjenu potpuno povezanih slojeva konvolucijskim slojevima tako da se interni signali dobivaju na način:

$$z_t = \sigma(\text{conv}_{3x3}([h_{t-1}, x_t], W_z)) \tag{4.4}$$

$$r_t = \sigma(\text{conv}_{3x3}([h_{t-1}, x_t], W_r)) \tag{4.5}$$

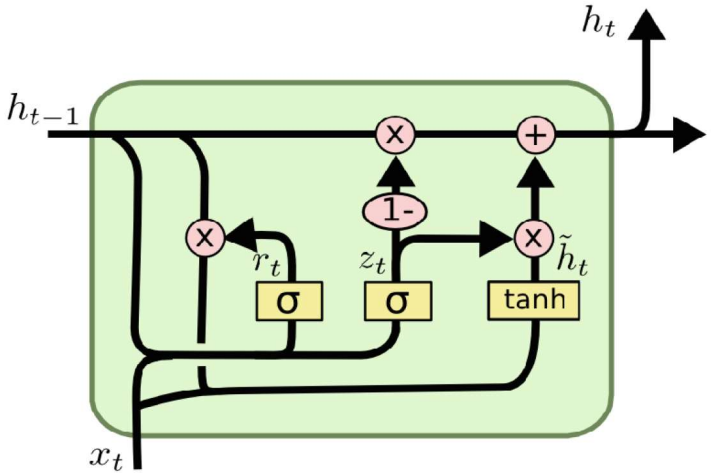
$$\widetilde{h_t} = \tanh(\text{conv}_{3x3}([r_t \odot h_{t-1}, x_t], W_h)) \tag{4.6}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \tag{4.7}$$

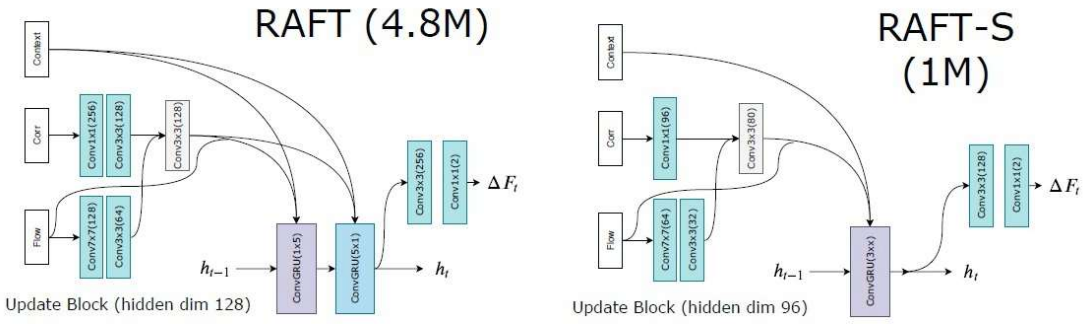
Gdje je x_t ulaz u GRU ćeliju čija će struktura biti objašnjena kasnije u opisu rada metode.

Ćelija GRU provodi ugađanje trenutne procjene toka te je zbog toga smatramo kao središnju komponentu operatra ažuriranja. Bitno svojstvo RAFT-a koje treba spomenuti je da su težine unutar GRU ćelije dijeljene tijekom svih iteracija ugađanja. Potpuna arhitektura operatora ažuriranja je prikazana na slici 4.6. U RAFT implementaciji sa 4.8 miliona parametara, autori zapravo predlažu dvije uzastopne GRU ćelije sa različitim konvolucijskim jezgrama u svojim internim slojevima. Ulaz x_t u ćeliju GRU je kombinacija informacija akumuliranih iz prethodnih iteracija. Broj ponavljanja ove procedure je hiperparametar modela i može utjecati na konačnu kvalitetu

procjene toka, no prevelik broj ponavljanja nema smisla budući da je operator ažuriranja dizajniran tako da vrijednosti ugađanja konvergiraju u nulu što znači da neće biti naknadnih ažuriranja vrijednosti toka.



Slika 4.5: Blok shema ćelije GRU (engl. gated recurrent unit). Ponašanje ćelije GRU određuju njezini interni signali r_t, z_t i $\widetilde{h}_t$. Signal z_t predstavlja vrata za ažuriranje, a signal r_t vrata za resetiranje. U izvedbi, to su vektori koji definiraju koje informacije će se propagirati prema izlazu. Ćeliju je moguće trenirati da pamti važne informacije neodređeno dugo, a odbacuje informacije koje nisu značajne za obradu. Vrata za ažuriranje i resetiranje se konstruiraju kombinacijom ulaza i unutarnjeg stanja sustava pomnoženih vlastitim težinama koje se razlikuju u oba slučaja. Izlaz ćelije h_t je definiran vratima ažuriranja koja određuju koliko izlazne informacije će biti korišteno od trenutnog sadržaja memorije, a koliko od stanja memorije ćelije iz prethodnog koraka. Na taj način GRU ćelija ima mogućnost pohrane i filtriranja informacija. Ispravnim postavljanjem vrijednosti pojedinih vrata možemo u potpunosti propagirati vrijedne informacije dobivene od prethodne ćelije te na taj način rješavamo problem nestajućeg gradijenta [24, 10]. Slika prikazuje tok podataka, a kružići predstavljaju operacije podacima koji ulaze u njih: $\times$ znači množenje, + je zbrajanje, a 1- je oduzimanje svih elemenata tenzora od jedinice tj. $f(x) = 1 - x$. **Izvor:** https://technopremium.com/blog/wp-content/uploads/2019/06/gru-1-1200x600.png



Slika 4.6: Arhitektura modula za ažuriranje. Broj i veličina slojeva ponovo ovise o inačici implementacije RAFT-a. Oba modela obrađuju trenutnu procjenu optičkog toka dvijema konvolucijskim slojevima s ReLU aktivacijom. Tako izlučene značajke se kombiniraju s mapama značajki dobivenim unaprijednim prolazom kroz slojeve za obradu korelacijskog vektora. Korelacijski vektor je rezultat operatora L (engl. lookup) koji će biti objašnjen kasnije, a služi za pridruživanje piksela odgovarajućem oknu volumena cijene. Operator ažuriranja kombinira informacije o trenutnoj procjeni toka i sličnosti parova piksela u odgovarajućem oknu dobivenog volumena cijene te nastoji pronaći najbolje podudarnosti pojedinih piksela. Opisana kombinacija trenutnog toka i tzv. lokalnog korelacijskog tenzora, dobivenog pridruživanjem piksela pripadajućem oknu volumena cijene, obrađuje se još jednim konvolucijskim slojem te zajedno sa mapama značajki konteksta dovodi na ulaz GRU ćelije kao ulazni signal x_t. Podsjetimo se, mape značajke konteksta dobivene su prethodno opisanom komponentom kodera značajki iz prve od dvaju slika dovedenih na ulaz kao par. Važna razlika RAFT-a i RAFTS-a je u internim slojevima ćelije GRU. RAFT model koristi dvije uzastopne GRU ćelije od kojih prva provodi konvoluciju jezgrom u obliku vektora dimenzija 1×5 orjentiranog horizontalno u prostoru mapi značajki, dok druga ćelija koristi jezgru dimenzija 5×1 orjentiranu vertikalno. RAFT-S model koristi jednu GRU ćeliju uz veličinu jezgre 3×3 u svojim internim konvolucijskim slojevima. Polje vektora kojim ažuriramo trenutnu procjenu optičkog toka, $\mathbf{\Delta f}$, dobivamo provlačenjem izlaza GRU ćelije kroz 2 konvolucijska sloja na izlazu operatora ažuriranja [43].

4.2. Detaljan opis metode

U ovom odjeljku ćemo opisati rad modela RAFT za procijenu optičkog toka. Detaljnije ćemo objasniti tok i strukturu podataka među slojevima i komponentama modela. Arhitekturu pojedinih komponenata smo opisali u prethodnom odjeljku dok ćemo u nastavku pobliže objasniti njihovo međudjelovanje.

Zatadak modela je pronaći funkcije preslikavanja ($\mathbf{f}^1$, $\mathbf{f}^2$) koje definiraju relativno kretanje svakog pojedinog piksela između slika I_1 i I_2. Korespondirajući piksel iz isječka I_1 tada u I_2 ima oblik: $I_2(u', v') = I_1(u, v)$ gdje je $(u', v') = (u + f^1(u, v), v + f^2(u, v))$. Navedena jednadžba vrijedi samo za piksele $(u, v) \in \mathfrak{D}(I_1)$ koji imaju pripadajuću korespondenciju u I_2.

4.2.1. Izlučivanje korespondencijskih značajki

Zadatak izlučivanja korespondencijskih značajki obavlja koder značajki opisan u prethodnom odjeljku. Na ulaz kodera dovodimo par slika I_1, I_2. Dimenzije ulaznog podatka su određene visinom i širinom slike te dubinom koja je u ovom slučaju jednaka tri budući da se svaka slika sastoji od 3 kanala boje, tj. $I_1, I_2 \in \mathbb{R}^{H \times W \times 3}$. Unaprijednim prolazom kroz slojeve kodera značajki dobivamo D mapa značajki, za svaku od dvije slike na ulazu. U slučaju RAFT-S izvedbe, D je postavljen na vrijednost 128, odnosno 256 u RAFT izvedbi. Prostorne dimenzije pojedinih mapa značajki su smanjene 8 puta u odnosu na izvorne dimenzije slika. Dakle, izlazne mape značajki $g_0(I_1)$ i $g_0(I_2)$ imaju oblik $\mathbb{R}^{H/8 \times W/8 \times D}$.

Mape značajki konteksta $h_0(I_1)$ imaju isti oblik, tj. $h_0(I_1) \in \mathbb{R}^{H/8 \times W/8 \times D}$. Autori eksperimentalno dokazuju da dodavanje značajki konteksta direktno u operator ažuriranja pridonosi poboljšanju rezultata u procjeni toka [43]. Taj rezultat možemo objasniti činjenicom da izvorne značajke prve slike pomažu u procjeni toka tako što unose informacije o slici na izvornoj visokoj rezoluciji. Također, pomažu u odlučivanju toka u zaklonjenim pikselima i zamućenim dijelovima slike.

Važno je napomenuti da se ovaj postupak izdvajanja značajki provodi samo jednom za dovedeni par ulaznih primjera te se dobivene značajke koriste tijekom svih iteracija algoritma. Arhitekturu kodera značajki prikazujemo na slici 4.3.

4.2.2. Određivanje sličnosti među svim parovima piksela

Sljedeći korak u modelu RAFT za procjenu optičkog toka je stvaranje korelacijskih polja koja će služiti za određivanje vizualnih sličnosti među pikselima uzastopnih slika.

Korištenjem podatkovnih struktura opisanih u prethodnom poglavlju, izgrađujemo 4-slojnu korelacijsku piramidu gdje svaki sloj sadržava volumen cijene dobiven sažimanjem prosječnom vrijednosti posljednje dvije dimenzije ovisno veličini jezgre. Dakle, podatkovna struktura se sastoji od korelacijskih slojeva $\{C^1, C^2, C^3, C^4\}$ gdje je svaki C^k polje 4 dimenzije oblika: $H \times W \times H/2^k \times W/2^k$. Primjeri izgradnje takvih polja su prikazani na slici 3.2. Elemente matrice C^k konstruiramo skalarnim umnoškom vizualnih značajki po trećoj dimenziji D dobivenih od kodera značajki.

Informacija o sličnosti piksela sa svakim njegovim mogućim parom iz druge slike je potrebna operatoru ažuriranja u svrhu određivanja najboljih mogućih korespondencija. Ta informacija o sličnostima parova je upravo dostupna u konstruiranoj korelacijskoj piramidi. Međutim, s obzirom da je RAFT iterativan model te se ukupna procjena toka odvija u koracima po principu malih pomaka prema konačnom rješenju, operator ažuriranja ne zahtjeva potpunu korelacijsku piramidu sa svim parovima, već mu je dovoljan isječak koji opisuje sličnosti piksela p_1 s lokalnim područjem oko piksela p_2 iz druge slike s kojim je p_1 trenutno uparen nakon n-te iteracije. Ako je trenutna procjena toka u nekoj iteraciji postupka jednaka (f^1, f^2). Trenutno uparivanje p_1 s p_2 definiramo kao:

$$p_2 = (p_{2u}, p_{2v}) = (p_{1u} + f^1(u, v), p_{1v} + f^2(u, v)) \tag{4.8}$$

Nakon toga definiramo lokalno područje radijusa r oko dobivenog piksela p_2 u I_2. Lokalno područje je skup slikovnih elemenata koji su za najviše r jedinica udaljeni od izvorišnog elementa p_2 korištenjem L1 mjere. Taj skup možemo zapisati:

$$N(p_2)_r = \{p_2' + dx \mid dx \in \mathbb{Z}^2, ||dx||_1 <= r\} \tag{4.9}$$

U iteraciji n+1, uparivanje piksela p_1 može potencijalno biti provedeno sa bilo kojim pikselom p_2' u lokalnom susjedstvu $N(p_2)_r$ oko p_2, ovisno o sličnosti izračunatoj u razinama korelacijske piramide. Na taj način, malim koracima koji očito mogu biti najveće vrijednosti $|r|$, operator ažuriranja konvergira prema optimalnoj procjeni toka dok je zadaća korelacijskog sloja prosljeđivanje informacija o sličnosti piksela iz I_1 s njemu pripadajućim lokalnim susjedstvom u I_2. Povezivanje piksela sa pripadajućim oknima volumena cijene zadatak je operatora pretraživanja L_c (engl. lookup operator). Uzimajući u obzir trenutnu procjenu toka, operator pretraživanja L_c generira lokalni korelacijski tenzor koji sadrži piksel p_1 iz prve slike uparen s oknom volumena cijene iz susjedstva p_2 dimenzije $2r \times 2r$, na razini piramide k. Zaključujemo da operator pretraživanja na ulazu prima slojeve korelacijske piramide tj. volumene cijene, a na

svom izlazu generira tenzor koji se prosljeđuje operatoru ažuriranja, a sadrži sličnosti svakog piksela sa pikselima iz susjedstva pomaknutog za trenutni iznos optičkog toka.

Domena i kodomena operatora pretraživanja L_c imaju oblik:

$$L_c : \{C^1, C^2, ..., C^k\} \mid C^k \in \mathbb{R}^{H \times W \times H/2^k \times W/2^k} \rightarrow \mathbb{R}^{H \times W \times k \times 2r \times 2r} \tag{4.10}$$

Pri tome je k broj razina u korelacijskoj piramidi. Uparivanja se provode na svim razinama korelacijske piramide čime postižemo efikasno pronalaženje korespondencija podjednako za fine i grube pomake.

Računanje korelacijskih polja svih parova piksela iz dvije uzastopne slike može biti računski zahtjevan posao. Ako je N broj piksela, tada računanje korelacijskog polja ima složenost $O(N^2)$, no s druge strane, taj posao je potrebno izvršiti samo jednom za određeni ulazni par. Za izbjegavanje velike računske složenosti, autori predlažu implementaciju za računanje korelacija među svim parovima koja ima složenost $O(NM)$ gdje je M broj iteracija koje postupak provodi [43]. U toj se implementaciji ne računaju korelacije za sve parove slikovnih elemenata unaprijed, nego se računanje obavlja na zahtjev prema formuli [43]:

$$C^m_{ijkl} = \langle g^{(1)}_{i,j}, \frac{1}{2^{2m}} (\sum_{p}^{2^m} \sum_{q}^{2^m} g^{(2)}_{2^m k+p, 2^m l+q}) \rangle \tag{4.11}$$

Pri tome je C^m_{ijkl} element volumena cijene na razini m, a $g^{(1)} = g_0(I_1)$ i $g^{(2)} = g_0(I_2)$.

Međutim, eksperimentalno se pokazalo da inicijalno računanje koralacija svih parova slikovnih elemenata nije računski preskupo u odnosu na ostatak postupka pa se može koristiti kao takvo.

4.2.3. Iterativno ugađanje toka

Kombiniranjem mapi značajki konteksta, lokalnog korelacijskog tenzora i trenutne procjene toka, provodimo iterativno ugađanje toka počevši od inicijalne vrijednosti toka. Uzastopnim iteracijama dobivamo niz procjena toka $\{\mathbf{f}_1, ..., \mathbf{f}_N\}$ te vrijedi:

$$\lim_{k \rightarrow \infty} \mathbf{f_k} = \mathbf{f}^* \tag{4.12}$$

gdje je $\mathbf{f}^*$ fiksna točka u koju postupak konvergira. Konvergencija postupka je osigurana dijeljenjem težina unutar operatora ažuriranja u svim iteracijama.

Za procjenu toka u slikovnim elementima koji nemaju korespondenciju u I_2, operator za ugađanje toka provodi interpolaciju na temelju najbližeg susjeda. To znači da se za iznos i smjer toka piksela, koji nema korespondenciju zbog zaklanjanja ili drugih razloga, kopira vrijednost toka njegovog najbližeg susjednog piksela za kojeg je određena pripadajuća korespondencija.

Da bi definirali tijek događaja unutar jedne iteracije, trebamo poznavati njezine glavne komponente. Centralna komponenta jedne iteracije je pravilo ažuriranja toka koji je inicijalno postavljen na određenu vrijednost prema odabranim pravilima. Rezultat iteracije ovisi u unutarnjem stanju operatora ažuriranja i ulaznim podacima koji su strogo strukturirani. U nastavku će biti opisana pravila pojedinih faza jedne iteracije.

- **Inicijalizacija**

  Provodi se prije prve iteracije postupka tako da za sve piksele postavljamo vrijednosti vektora optičkog toka na nul-vektore. Kod procjene optičkog toka na uzastopnim povezanim parovima ulaza kao što je slučaj u videu, za inicijalne vrijednosti možemo iskoristiti procijenjeni tok prethodnog para. Nadamo se da ćemo time već u početku biti bliže ispravnoj procjeni budući da su parovi uzastopno vezani i pomaknuti za vrlo mali vremenski trenutak te zbog toga nema velikih razlika u sljedećem okviru (engl. frame) videa.

- **Ulaz**

  Ulazni podatak u svaku iteraciju algoritma predstavlja mapa značajki stvorena kao kombinacija trenutne procjene toka, značajki konteksta i lokalnog korelacijskog tenzora kao što je objašnjeno ranije.

- **Ažuriranje toka**

  Ažuriranje trenutne procjene optičkog toka odvija se korištenjem ćelije GRU čiju smo arhitekturu opisali u prethodnom poglavlju. Svaka iteracija postupka, uz trenutnu procjenu toka, ažurira i varijablu h unutarnjeg stanja sustava koja se koristi u narednim iteracijama. Arhitekturu i unutarnje signale ćelije GRU prikazujemo na slici 4.5.

- **Izlaz**

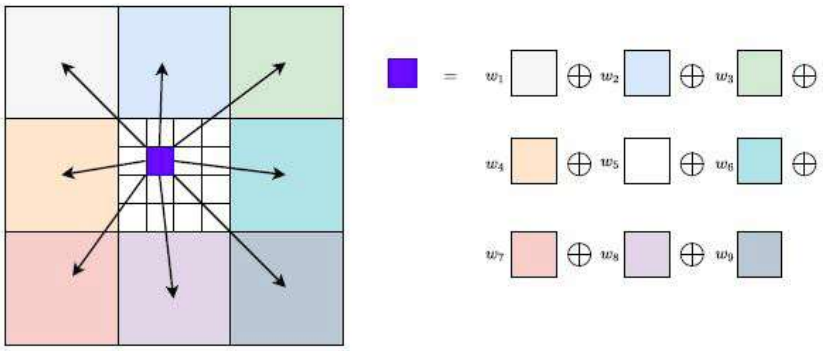
  Važno je spomenuti da operator ažuriranja u svim iteracijama radi sa tokom na rezoluciji 8 puta manjoj po svakoj prostornoj dimenziji u odnosu na izvorne rezolucije slika. To je posljedica arhitekture kodera značajki koji je generirao mape značajki na 8 puta manjoj rezoluciji od izvorne. Konačan izlaz iteracije je dobiven provlačenjem rezultata GRU ćelija kroz dvoslojnu konvolucijsku mrežu koja određuje masku težina susjeda koje će biti korištene u postupku

skaliranja. Izlaz jedne iteracije predstavlja vektor pomaka kojim ažuriramo trenutnu procjenu optičkog toka. Prije prosljeđivanja procijenjenog toka na izlaz modela, potrebno je izlaz operatora ažuriranja skalirati na izvornu rezoluciju slika korištenjem spomenute naučene maske težina.

Slika 7.7 vizualizira tijek ugađanja optičkog toka uzastopnim iteracijama.

4.2.4. Skaliranje toka na izvornu rezoluciju

Nakon završetka ugađanja procjene toka, nužno je povećanje rezolucije kako bi dobiveni rezultat bio usporediv sa označenim primjerima. Povećanje rezolucije provodi se tako da vrijednost optičkog toka svakog piksela u visokoj rezoluciji dobivamo korištenjem softmax funkcije nad težinskom kombinacijom susjednih 9 vrijednosti tokova. Međutim, susjednosti su definirane na radnoj, niskoj, rezoluciji. Težine pojedinih susjeda su određene konvolucijskom mrežom kako bismo postigli optimalnu kombinaciju susjednih elemenata, a za njihovu pohranu koristimo spomenutu masku težina. Opisani postupak skaliranja toka na izvornu rezoluciju je računski vrlo složena operacija. Postupak povećanja rezolucije prikazujemo na slici 4.7.



Slika 4.7: Postupak povećanja rezolucije konveksnom kombinacijom susjeda u izvornoj rezoluciji [43].

4.2.5. Funkcija gubitka

Tjekom faze nadziranog učenja modela, koristimo funkciju gubitka koja predstavlja L1 udaljenost između procijenjenog i stvarnog optičkog toka. Tijekom N iteracija algoritma kao što je opisano u prethodnim odlomcima, dobivamo niz procjena optičkog toka $\{\mathbf{f}_1, ..., \mathbf{f}_N\}$ gdje $\mathbf{f}_k$ predstavlja procjenjeni optički tok u k-toj iteraciji. Funkcija gubitka je definirana kao:

$$L = \sum_{i=1}^{N} \gamma^{N-i} ||\mathbf{f}_{gt} - \mathbf{f}_i||_1 \tag{4.13}$$

U prikazanoj jednadžbi $\mathbf{f}_{gt}$ označava ručno označeni točni (eng. ground truth) tok, a γ parametar težine koju eksponencijalno povećavamo pridjeljivajući veće težine višim iteracijama. Time više kažnjavamo grešku koja nastaje u kasnijim iteracijama postupka. Ugađanje na posljednjim iteracijama ima izravan utjecaj na tok koji se vraća kao izlaz modela. Greške koje se javljaju u toj fazi trebaju imati veću težinu od grešaka u početnim iteracijama gdje se očekuje da tok još nije ispravno procjenjen. Autori predlažu postavljanje parametra na $\gamma = 0.8$ tijekom eksperimenata [43].

4.3. Usporedba RAFT-a s drugim modelima

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)
		Clean	Final	F1-epe	F1-all	Clean	Final	F1-all
-	FlowFields	-	-	-	-	3.75	5.81	15.31
-	FlowFields++	-	-	-	-	2.94	5.49	14.82
S	DCFlow	-	-	-	-	3.54	5.12	14.86
S	MRFlow	-	-	-	-	2.53	5.38	12.19
C + T	HD3	3.84	8.77	13.17	24.0	-	-	-
	LiteFlowNet	2.48	4.04	10.39	28.5	-	-	-
	PWC-Net	2.55	3.93	10.35	33.7	-	-	-
	LiteFlowNet2	2.24	3.78	8.97	25.9	-	-	-
	VCN	2.21	3.68	8.36	25.1	-	-	-
	MaskFlowNet	2.25	3.61	-	23.1	-	-	-
	FlowNet2	2.02	3.54[1]	10.08	30.0	3.96	6.02	-
	Ours (small)	2.21	3.35	7.51	26.9	-	-	-
	Ours (2-view)	**1.43**	**2.71**	**5.04**	**17.4**	-	-	-
C+T+S/K	FlowNet2	(1.45)	(2.01)	(2.30)	(6.8)	4.16	5.74	11.48
	HD3	(1.87)	(1.17)	(1.31)	(4.1)	4.79	4.67	6.55
	IRR-PWC	(1.92)	(2.51)	(1.63)	(5.3)	3.84	4.58	7.65
	ScopeFlow	-	-	-	-	3.59	4.10	6.82
	Ours (2-view)	(0.77)	(1.20)	(0.64)	(1.5)	**2.08**	**3.41**	**5.27**
C+T+S+K+H	LiteFlowNet2[2]	(1.30)	(1.62)	(1.47)	(4.8)	3.48	4.69	7.74
	PWC-Net+	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72
	VCN	(1.66)	(2.24)	(1.16)	(4.1)	2.81	4.40	6.30
	MaskFlowNet	-	-	-	-	2.52	4.17	6.10
	Ours (2-view)	(0.76)	(1.22)	(0.63)	(1.5)	1.94	3.18	**5.10**
	Ours (warm-start)	(0.77)	(1.27)	-	-	**1.61**	**2.86**	-

Slika 4.8: Rezultati evaluacije modela na podatkovnim skupovima Sintel i KITTI uz različite skupove korištene za učenje i fino ugađanje modela. Kod evaluacije skupa KITTI, F1-epe mjera predstavlja srednju euklidsku udaljenost procijenjenog i stvarnog optičkog toka u pikselima dok F1-all mjera predstavlja postotak pogrešno procjenjenih piksela. Prvi odjeljak tablice prikazuje rezultate evaluacije nekih postojećih modela treniranim na podatkovnom skupu Sintel. Drugi odjeljak tablice uspoređuje rezultate evaluacije RAFT modela sa drugim modelima uz korištenje skupova Flying Chairs i Flying Things u fazi učenja. Treći odjeljak tablice prikazuje rezultate evalucije modela uz fino ugađanje na skupu Sintel odnosno KITTI. Posljednji, četvrti, odjeljak tablice prikazuje rezultate evaluacije modela uz korištenje kombinacije skupova Sintel, KITTI i HD1K u fazi finog ugađanja modela. Vidimo da upravo u tom slučaju RAFT model postiže najbolje rezultate evaluacije na promatranim skupovima. Dakle, možemo zaključiti da je korištenje više različitih podatkovnih skupova u fazi finog ugađanja pridonijelo boljoj generalizaciji pri procjeni optičkog toka [43].

5. Nenadzirano učenje RAFT-a

U ovom poglavlju baviti ćemo se implementacijom samonadziranog postupka učenja RAFT modela za optički tok. Iako je RAFT originalno namijenjen učenju optičkog toka na označenim podacima, predmet istraživanja ovog rada je mogućnost nenadziranog učenja RAFT-a te usporedba rezultata s ostalim nenadziranim kao i nadziranim modelima. Za početak ćemo pobliže definirati pojam nenadziranog učenja, a zatim detaljno opisati naš postupak nenadziranog učenja koji uključuje transformiranje podataka na kojima se uči, promjene u originalnom modelu, formiranje funkcija gubitaka i optimizacijski postupak uz unatražni prolaz. Implementaciju postupaka izvodimo u programskom okviru PyTorch te podrazumijevamo poznavanje osnovnih metoda sadržanih u programskom paketu. Svaka pojedina klasa i metoda PyTorch paketa je detaljno opisana u javno dostupnoj dokumentaciji te će se kao takve koristiti u isječcima programskog koda bez objašnjenja ponašanja.

5.1. Nenadzirano učenje

Nenadzirano učenje je vrsta strojnog učenja u kojem modelu ne prikazujemo oznake koje predstavljaju željeni izlaz uz dovedene pripadajuće podatke na ulaz [13]. Ispravno formulirani postupak učenja dovodi model u stanje u kojem je model sposoban generirati željene oznake iako ih tijekom učenja nije vidio. Cilj nenadziranog učenja jest pronaći pravilnosti u podacima koje određuju ponašanje modela u skladu sa zadanom funkcijom cilja. Tipični problemi koji se rješavaju nenadziranim učenjem su grupiranje podataka i smanjenje dimenzionalnosti. Navedena se paradigma učenja može koristiti za složenije probleme, ovisno o domeni primjene i traženoj razini točnosti. Pravilnosti u podacima mogu biti implicitno naučene uz rješavanje zamjenskog zadatka (engl. pretext task) i iskorištene za rješavanje glavnog zadatka. Na taj način generiramo oznake za učenje direktno iz podataka i učimo na tim oznakama pa govorimo o samonadziranom učenju. Samonadzirano učenje (engl. self-supervision learning) je tehnika u kojoj model uči na oznakama koje je sam prethodno generirao. Samonadzirano učenje je

podskup nenadziranog učenja jer i dalje ne koristi točne oznake.

Predmet istraživanja ovog rada je učenje modela za procjenu optičkog toka bez poznavanja točnog optičkog toka u slikama za učenje.

5.2. Transformacije na skupu podataka

Procjena optičkog toka je zadatak koji želimo dobro obaviti i na novim primjerima koji nisu viđeni tijekom učenja, bili oni označeni ili ne. Učenje na konačnom skupu podataka, koji predstavlja podskup beskonačnog skupa stvarnih podataka, unosi rizik od prenaučenosti (engl. overfiting). Prenaučenost je svojstvo modela da postiže izvrsne rezultate na podacima sa kojima je učeno, dok podbacuje na novim podacima, a uzrok tome je prilagođavanje modela specifičnom šumu u podacima za učenje. Za takve modele kažemo da imaju loše svojstvo generalizacije. Dobro generalizirati znači postizati zadovoljavajuću točnost u radu s podacima koji nisu viđeni tijekom faze učenja.

U našem slučaju, ulazni podaci su u obliku parova slika. Prilagođavanje modela nekim specifičnim uzorcima koji se pojavljuju u ulaznim slikama može značiti lošu generalizaciju. Iz tog razloga, želimo tijekom faze učenja modelu prikazati što je moguće više različitih slika kojima ćemo simulirati moguće varijacije podataka odnosno šumove koji se javljaju u praksi. Budući da je prostor mogućih ulaznih parova slika beskonačan, jasno je da ne možemo modelu prikazati sve parove koji se mogu pojaviti već raspolažemo sa konačnim skupom parova slika za učenje. Kako bi taj konačni skup proširili, provodimo nasumičnu transformaciju dostupnih parova slika za učenje.

Transformacije koje provodimo su:

- **Prostorna transformacija**
  Prostorna transformacija podrazumijeva skaliranje i obrezivanje slike. Ulazni par slika simetrično skaliramo nasumičnim faktorom u definiranom intervalu te izrežemo nasumični dio slika dimenzija određenima parametrom veličine obrezivanja. Zbog ograničenih resursa, postupak učenja provodimo uz postavke parametra veličine obrezivanja prikazane u tablici 5.1.

Podatkovni skup	Originalna veličina slika	Obrezana veličina
FlyingChairs	384×512	368×496
FlyingThings	540×960	312×624
Sintel	436×1024	304×608
Cityscapes	1024×2048	304×608

Tablica 5.1: Postavke parametara obrezivanja tijekom faze učenja.

– **Fotometrijska transformacija**

Fotometrijsku transformaciju provodimo korištenjem klase *ColorJitter* iz PyTorchevog paketa *torchvision.transforms*. Pri kreiranju objekta, potrebno je definirati parametre nasumičnih transformacija slika. Parametrima zadajemo interval promjene svjetline, kontrasta, zasićenja i nijanse boje u slici. Svaka transformacija je nasumična i izvodi se tako da se svaki od navedenih parametara slike postavlja na nasumičnu vrijednost iz definiranog intervala. Budući da ulaz u naš model predstavlja par slika, transformaciju je moguće učiti simetrično ili asimetrično. Kod simetrične transformacije, nasumično odabrani parovi se primjenjuju na par slika zajedno, dok se kod asimetrične transformacije parametri nasumično biraju za svaku sliku posebno. Kreiranje objekta klase *ColorJitter* i definiciju odgovarajuće metode za transformiranje prikazujemo na slici 5.1.

```
    # photometric augmentation params
    self.photo_aug = ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.5/3.14)

def color_transform(self, img1, img2):
    """ Photometric augmentation """

    # asymmetric
    if np.random.rand() < self.asymmetric_color_aug_prob:
        img1 = np.array(self.photo_aug(Image.fromarray(img1)), dtype=np.uint8)
        img2 = np.array(self.photo_aug(Image.fromarray(img2)), dtype=np.uint8)

    # symmetric
    else:
        image_stack = np.concatenate([img1, img2], axis=0)
        image_stack = np.array(self.photo_aug(Image.fromarray(image_stack)), dtype=np.uint8)
        img1, img2 = np.split(image_stack, 2, axis=0)

    return img1, img2
```

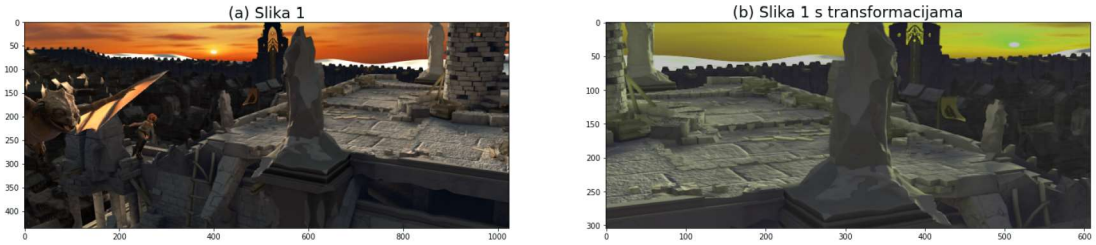
Slika 5.1: Kreiranje objekta klase *ColorJitter* i definicija metode za provođenje fotometrijske transformacije. Vjerojatnost provođenja asimetrične odnosno simetrične transformacije je određena parametrom.

– **Zrcaljenje**

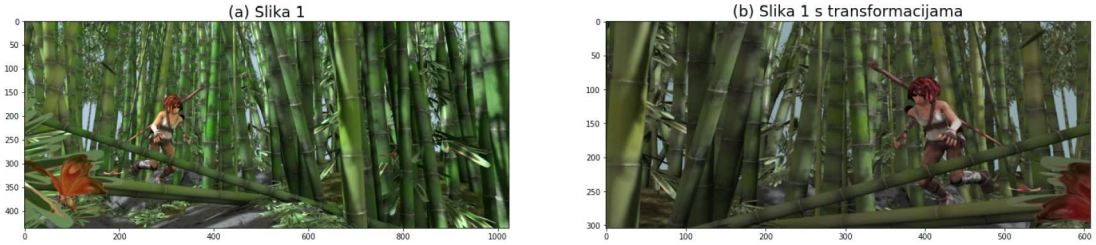
Uz određenu vjerojatnost koju definiramo kao hiperparametar postupka, ulazne slike zrcalimo oko horizontalne odnosno vertikalne osi. Zrcaljenjem postižemo veću raznolikost u skupu za učenje te očekujemo bolje svojstvo generalizacije modela.

Primjere nasumično provedenih transformacija prikazujemo na slikama 5.2, 5.3 i 5.4.

Slika 5.2: Primjer fotometrijske transformacije.



Slika 5.3: Primjer fotometrijske transformacije.



Slika 5.4: Primjer fotometrijske transformacije.

5.3. Izmjene u modelu RAFT

Budući da je RAFT izvorno namijenjen nadziranom učenju optičkog toka, arhitektura modela je građena s ciljem da podrži tu ideju i postigne što bolje rezultate na skupovima za evaluaciju. Detalji u izvedbi modela za nadzirano odnosno nenadzirano učenje mogu se razlikovati. Autori rada What Matters in Unsupervised Optical Flow [22] analiziraju upravo te razlike i predlažu tehnike poboljšanja rezultata u postupcima nenadziranog učenja. Tehnike se odnose na promjenu ponašanja tijekom unatražnog prolaza, načina dobivanja toka, načina upravljanja zaklonjenim područjima, obradu i transformaciju ulaznih podataka itd.

Nama najzanimljivija tehnika za poboljšanje rezultata nenadziranog postupka koju predlažu autori je normalizacija volumena cijene (engl. cost volume normalization). Originalna implementacija RAFT-a ne uključuje navedenu normalizaciju volumena cijene te trebamo učiniti promjene u komponenti korelacijskog sloja modela. Normalizaciju volumena cijene vršimo prema formuli iz [22]:

$$C^{(l)}_{x,y,u,v} = \sum_{d} \left(\frac{F^{(1,l)}_{x,y,d} - \mu^{(1,l)}}{\sigma^{(1,l)}} \right) \left(\frac{F^{(2,l)}_{x+u,y+v,d} - \mu^{(2,l)}}{\sigma^{(2,l)}} \right) \tag{5.1}$$

Pri tome je $F^{(i,l)} \in \mathbb{R}^{\frac{H}{2^l} \times \frac{W}{2^l} \times D}$ skup mapa značajki dubine D slike i na razini l korelacijke piramide. Formula 5.1 govori da normalizaciju treba provesti za svaki piksel (x, y) i sve moguće pomake (u, v), tj. sve moguće parove piksela odnosno cijeli volumen cijene. U (5.1), $\mu^{(i,l)}$ i $\sigma^{(i,l)}$ predstavljaju srednju vrijednost i standardnu devijaciju mape $F^{(i,l)}$ po svim dimenzijama.

Osim normalizacije volumena cijene, mijenjamo i normalizacijske slojeve kodera značajki i to iz normalizacije po podatcima (eng. batchnorm) u normalizaciju po kanalima (eng. gropnorm). Ova promjena doprinosi poboljšanju rezultata kod postupka učenja na malim grupama (engl. mini-batch). Također, ovdje ne koristimo asimetrične fotometrijske transformacije iz slike 5.1. za razliku od originalnog prijedloga autora RAFT-a. Razlog tome je funkcija gubitka koju koristimo za nenadzirano učenje, a temelji se na razlici intenziteta pojedinih piksela. Ako piksele transformiramo na različite načine, nije ih moguće upariti na temelju sličnosti u intenzitetu. Zbog toga transformaciju moramo obavljati simetrično.

5.4. Postupak učenja

Nakon provedenih izmjena u arhitekturi RAFT modela i dohvaćanja parova slika iz podatkovnog skupa uz definirane transformacije, prelazimo na definiranje postupka nenadziranog učenja. Prema prijedlogu autora SelFlow-a [28], nenadzirano učenje obavljamo u dvije faze. U prvoj fazi radimo s modelom učiteljem tzv. NOC modelom (engl. non-occluded model) koji procjenjuje optički tok u nezaklonjenim područjima na temelju uparivanja fotometrijski konzistentnih intenziteta piksela. Nakon toga, u drugoj fazi, koristimo naučene predikcije modela učitelja za vođenje procesa učenja modela učenika tzv. OCC modela (engl. occluded model). Vođenje se odvija u umjetno zaklonjenim područjima tako da se predikcija NOC modela koristi kao točna oznaka za učenje OCC modela. Umjetnim zaklanjanjem postižemo bolju generalizaciju u zaklonjenim pikselima iz podatkovnog skupa. Prema prijedlogu iz literature [22, 14], svaki

fotometrijski gubitak temeljimo na cenzuskoj metrici. Cenzuska metrika dvaju okana odgovara Hammingovoj udaljenosti binarnih cenzuskih opisnika. Cenzuski opisnik okna $k \times k$ je binarni vektor duljine $k * (k-1)$ čiji elementi kazuju je li odgovarajući piksel okna veći ili manji od središnjeg piksela. U nastavku će biti opisan postupak census transformacije, a zatim tehnika upravljanja zaklonjenim područjima. Nakon toga ćemo prikazati kompletan postupak učenja. Za početak, definirajmo oznake zbog lakšeg praćenja postupka.

5.4.1. Notacijska konvencija

Parove slika dovedene na ulaz modela označujemo s I_1 i I_2. Svaka slika sadrži 3 kanala boje budući da su slike u RGB formatu. Zbog toga možemo pisati $I_1, I_2 \in \mathbb{R}^{H \times W \times 3}$, gdje su H i W visina odnosno širina slike. Oznaka $\mathbf{f}_{i \to j}$ predstavlja optički tok između slika I_i i I_j. U našem primjeru, tok između dovedenog para slika I_1 i I_2 označujemo s $\mathbf{f}_{1 \to 2}$. Tok $\mathbf{f}_{2 \to 1}$ ćemo zvati povratni tok budući da je dobiven zamjenom slika na ulazu. Uz poznati tok $\mathbf{f}_{i \to j}$ i pripadajući par slika, izobličavanje slike I_j korištenjem toka $\mathbf{f}_{i \to j}$ označavamo s $I^f_{j \to i}$. Navedeno izobličavanje je unatražno [37]. Informacije o zaklonjenim pikselima pohranjujemo u obliku mape zaklanjanja $O_{i \to j}$ koja predstavlja zaklonjena područja između slika I_i i I_j. Mapa zaklanjanja $O_{i \to j}$ se sastoji samo od vrijednosti 0 ili 1 gdje vrijednost 1 znači da pripadni piksel iz I_i nije vidljiv u I_j tj. kažemo da je piksel zaklonjen. Vrijednost 0 govori da nema zaklanjanja u pripadnom pikselu.

Kod učenja OCC modela, uvodimo novu oznaku $\widetilde{I}_2$ koja predstavlja sliku 2 iz ulaznog para na koju smo dodali umjetno zaklonjena područja. U tom slučaju, na ulazu modela imamo par I_1, $\widetilde{I}_2$, a pripadajući tok, masku zaklanjanja i izobličenu sliku označujemo s $\widetilde{\mathbf{f}}_{1 \to 2}$, $\widetilde{O}_{1 \to 2}$ i $\widetilde{I}^f_{2 \to 1}$.

5.4.2. Cenzuska transformacija

Census transformacija kodira svaki piksel u binarni niz koji se naziva cenzuski opisnik. Kodiranje se vrši na temelju usporedbe intenziteta piksela sa svim njegovim susjedima u 3×3 susjedstvu. Census potpis je duljine 8 bitova, te predstavlja informaciju o tome je li piksel većeg ili manjeg intenziteta u odnosu na svoje susjede. Census transformacija je vrlo popularna u širokom spektru postupaka koji uključuju rad sa slikama zbog robusnosti prema promjenama u osvjetljenju. Transformacijom izlučujemo lokalne odnose piksela okna, ne promatrajući apsolutne iznose njihovih intenziteta.

Neka je $I(x, y)$ funkcija koja opisuje vrijednosti intenziteta sive slike na mjestu piksela (x, y). Svaki od elemenata census potpisa se odnosi na jednog od 8 susjeda piksela, a kompletan opisnik dobivamo prema izrazu:

$$CT(x, y) = \bigotimes_{i=-n'}^{i=n'} \bigotimes_{j=-m'}^{j=m'} H(I(x + i, y + j) - I(x, y)) \tag{5.2}$$

Pri tome $\bigotimes$ predstavlja operator konkatenacije koji djeluje po cijelom susjedstvu piksela (x, y), $CT(x, y)$ potpis dujine 8 bitova, a $H : \mathbb{R} \to \{0, 1\}$ Heavisideova step funkcija.

$$H(x) = \begin{cases} 0 \; ako \; x < 0 \\ 1 \; ako \; x \geq 0 \end{cases} \tag{5.3}$$

Navedena transformacija se još naziva binarna census transformacija budući da se svaki susjed kodira u jednu od dvije mogućnosti. Osim toga, postoji ternarna cenzuska transformacija koja svakog susjeda piksela kodira u neku od tri mogućnosti te se kodiranja svih susjeda konkateniraju u potpis na isti način. Kodiranu riječ dobivenu transformacijom nazivamo cenzuski opisnik. Dakle, cenzuska transformacija sliku $C \times H \times W$ pretvara u tenzor $8 \times H \times W$ koji sadrži cenzuske opisnike. Ternarnu transformaciju računamo prema izrazu:

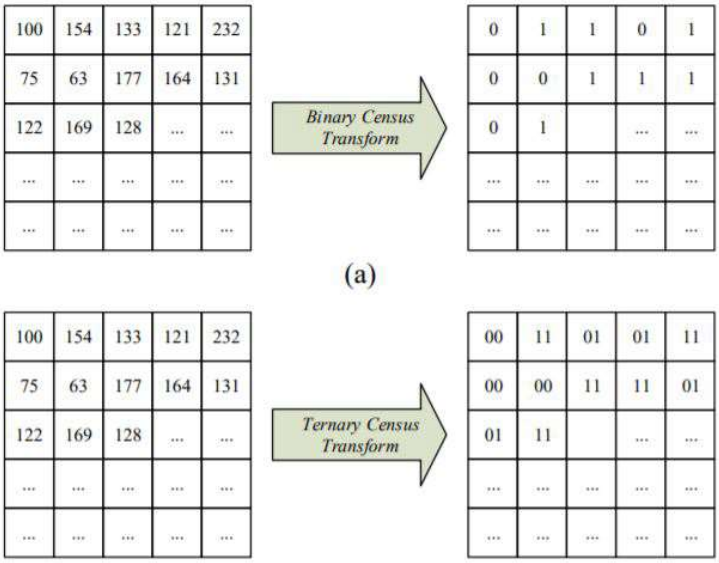
$$CT(x, y) = \bigotimes_{i=-n'}^{i=n'} \bigotimes_{j=-m'}^{j=m'} t(I(x + i, y + j), I(x, y)) \tag{5.4}$$

Gdje je t funkcija kodiranja oblika:

$$t(x, y, \delta) = \begin{cases} 00 \; ako \; x > y + \delta \\ 01 \; ako \; y - \delta \leq x \leq y + \delta \\ 11 \; ako \; x < y - \delta \end{cases} \tag{5.5}$$

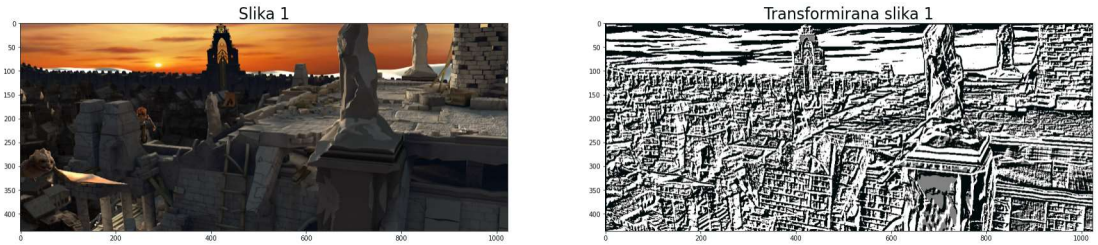
Nedostatak binarne cenzuske transformacije je osjetljivost na male promjene u intenzitetu piksela na područjima bez tekstura. Zbog toga, transformacija razlikuje piksele sličnih intenziteta iako pripadaju istoj površini zbog male promjene u intenzitetu. Ternarna transformacija rješava taj problem uvođenjem praga tolerancije δ u odstupanju intenziteta osvjetljenja. Drugi nedostatak je nediferencijabilnost cenzuskih metrika. Zbog tog svojstva, u implementacijama koristimo aproksimacije cenzuskih transformacija. Slika 5.5 prikazuje načelo generiranja potpisa transformacije u oba slučaja,

dok slike 5.6 i 5.7 prikazuju primjer provedene transformacije na slici iz podatkovnog skupa Sintel .



Slika 5.5: Postupak provođenja binarne odnosno ternarne census transformacije na mjestu centralnog piksela uz susjedstvo veličine 5×5. Slika je preuzeta iz [17].

Slika 5.6: Primjer provođenja ternarne census transformacije. Slika prikazuje jedan kanal dobivene transformacije, dok broj kanala ukupne transformacije sadrži broj kanala ovisan o broju piksela u susjedstvu.



Slika 5.7: Primjer provođenja ternarne census transformacije. Slika prikazuje jedan kanal dobivene transformacije, dok broj kanala ukupne transformacije sadrži broj kanala ovisan o broju piksela u susjedstvu.

5.4.3. Procjena zaklonjenih dijelova

Fotometrijski gubitak, koji će biti temelj našeg nenadziranog postupka, temelji se na očuvanju intenziteta piksela u obje slike iz para. Ukoliko piksel nema odgovarajućeg para, tvrdoglavo traženje sličnih piksela za uparivanje može navesti na krivi trag. Za takve piksele kažemo da su zaklonjeni u drugoj slici i potrebno je razviti mehanizam za njihovu detekciju i obradu.

Prijedlozi tehnike za detekciju zaklonjenih piksela iz literature [22, 28] podrazumijevaju poznavanje unaprijednog i povratnog toka. Te tokove možemo dobiti dvostrukim unaprijednim prolazom uz različit poredak ulaznih slika. Neka je $\mathbf{f}_{1\to2}$ unaprijedni tok između slika I_1 i I_2, a $\mathbf{f}_{2\to1}$ povratni tok dobiven zamjenom ulaznih slika, tj. između slika I_2 i I_1. Provjera konzistentnosti unaprijednog i unazadnog toka otkriva zaklonjena područja, a u tu svrhu trebamo izračunati obrnuti unaprijedni tok [28] prema:

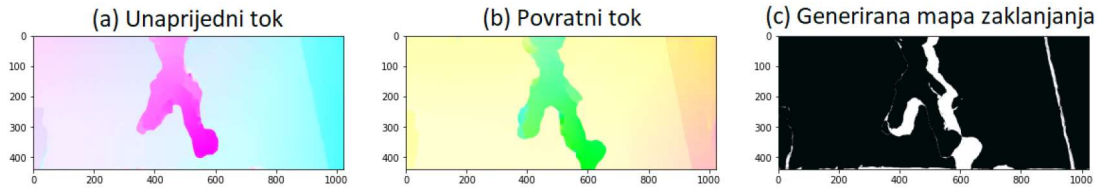
$$\hat{\mathbf{f}}_{1\to2} = -\mathbf{f}_{2\to1}(\mathbf{p} + \mathbf{f}_{1\to2}(\mathbf{p})) \tag{5.6}$$

Ako je unaprijedni tok konzistentan izračunatom obrnutom toku u nekom pikselu, kažemo da piksel nije zaklonjen. Provjeru konzistentnosti vršimo prema sljedećoj nejednakosti:

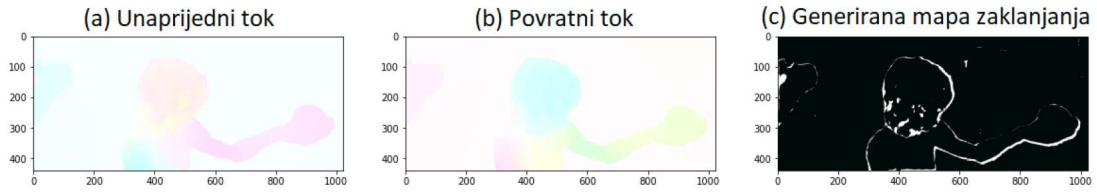
$$|\,\mathbf{f}_{1\to2} + \hat{\mathbf{f}}_{1\to2}\,|^2 < \alpha_1(|\,\mathbf{f}_{1\to2}\,|^2 + |\,\hat{\mathbf{f}}_{1\to2}\,|^2) + \alpha_2 \tag{5.7}$$

U eksperimentima postavljamo $\alpha_1 = 0.01$ i $\alpha_2 = 0.5$. Ako je nejednakost narušena, piksel smatramo zaklonjenim i odgovarajući element mape zaklanjanja $O_{1\to2}$ postavljamo na vrijednost 1, a inače na vrijednost 0. Analogno se računaju mape zaklanjanja za proizvoljni par ulaznih slika. Slika 5.10 prikazuje implementaciju metode koja provodi navedenu provjeru konzistentnosti i generira mapu zaklanjanja u

programskom jeziku Python. Na slikama 5.8 i 5.9 prikazujemo rezultat generiranja mape zaklanjanja na temelju provjere konzistentnosti unaprijednog i povratnog optičkog toka.



Slika 5.8: Primjer generiranja mape zaklanjanja na temelju dostupnog unaprijednog i povratnog optičkog toka.



Slika 5.9: Primjer generiranja mape zaklanjanja na temelju dostupnog unaprijednog i povratnog optičkog toka.

```python
def length_sq(x):
    return torch.sum(torch.square(x), dim=1, keepdim=True)


def get_occlusion(flow_fw, flow_bw):
    x_shape = flow_fw.shape

    flow_bw_warped = warp_utils.flow_warp(flow_bw, flow_fw)
    flow_diff_fw = flow_fw + flow_bw_warped
    mag_sq_fw = length_sq(flow_fw) + length_sq(flow_bw_warped)
    occ_thresh_fw = 0.01 * mag_sq_fw + 0.5
    occ_fw = (length_sq(flow_diff_fw) > occ_thresh_fw).type(torch.float)

    return occ_fw
```

Slika 5.10: Implementacija metode za generiranje mape zaklanjanja temeljene na provjeri konzistentnosti unaprijednog i povratnog optičkog toka.

5.4.4. Učenje NOC modela

Kao što je ranije objašnjeno, model učitelj nastoji naučiti procjenjivati optički tok u pikselima koji nisu zaklonjeni. Učenjem nastojimo minimizirati fotometrijski gubitak između originalne slike i slike dobivene izobličavanjem na temelju procjenjenog optičkog toka. Područja koja smo detektirali kao zaklonjena isključujemo iz postupka

računanja fotometrijskog gubitka budući da nema smisla kažnjavati piksele koji nemaju svoje parove.

Nakon provedenih transformacija na skupu za učenje objašnjenih u odjeljku 5.2, provodimo dva unaprijedna prolaza kroz duboki model uz različit poredak ulaznog para slika I_1 i I_2. Na taj način procjenjujemo unaprijedni i povratni optički tok. Na temelju procjenjenih tokova računamo vrijednost funkcije gubitka s obzirom na koju kasnije provodimo unatražni prolaz i optimiramo parametre modela. Funkcija gubitka temelji se na fotometrijskom gubitku uz zanemarivanje zaklonjenih područja. Za fotometrijski gubitak odabiremo Charbonnierov gubitak kojem kao argument predajemo Hammingovu udaljenost ternarnog cenzus opisnika slike 1 i opisnika izobličene slike generiranih prema (5.4). Funkcija gubitka ima oblik:

$$L_{sp} = \sum_{i=1}^{N} \gamma^{N-i} \frac{\sum \psi(D_H(CT(I_1), CT(I_{2\to1}^{f^i}))) \odot (1 - O_{1\to2})}{\sum(1 - O_{1\to2})} \tag{5.8}$$

Pri tome je $\psi(x) = (\mid x \mid^2 + \epsilon)^q$ funkcija charbonnierovog gubitka, $D_H(x, y)$ funkcija Hammingove udaljenosti dvaju binarnih vektora, a $CT(I)$ funkcija ternarne cenzuske transformacije koja provodi operaciju prema (5.4). Operator $\odot$ predstavlja množenje matrica po elementima. U eksperimentima vrijednosti parametara postavljamo na $q = 0.5$, $\epsilon = 0.001$ i $\gamma = 0.8$. Budući da je RAFT iterativan model, procjenu toka možemo evaluirati u svakoj iteraciji ugađanja. Zbog toga parametar težine γ eksponencijalno povećavamo pridjeljivajući veće težine gubitku u višim iteracijama. Time više kažnjavamo grešku koja nastaje u višim iteracijama postupka. Definiranu funkciju L_{sp} (5.8) možemo nazvati fotometrijskim gubitkom slijeda (engl. sequence photoloss function). Implementaciju metode za izračun spomenute funkcije gubitka u programskom jeziku Python prikazujemo na slici 5.11.

Slika 5.12 prikazuje standardnu strukturu poziva metoda prilikom treniranja dubokih modela u programskom okviru Pytorch. Nakon spomenuta dva unaprijedna prolaza i poziva funkcije gubitka, provodimo unatražni prolaz uz ograničavanje gradijenata na interval $[-1, 1]$ te optimizaciju postupkom AdamW [29].

Tehnike izobličavanja slike i filtriranja zaklonjenih područja koje koristimo u brojniku (5.8) prikazujemo na slici 5.13.

```python
def sequence_photoloss(flow_preds_for, flow_preds_bac, image1, image2, gamma=0.8):
    """ Loss function defined over sequence of flow predictions """

    n_predictions = len(flow_preds_for)
    flow_loss = 0.0

    for i in range(n_predictions):
        i_weight = gamma**(n_predictions - i - 1)

        warpedimg, occmap = warp_utils.flow_warp(image2, flow_preds_for[i], flow_preds_bac[i])

        diff = utils.hamming_distance(utils.ternary_transform(image1), utils.ternary_transform(warpedimg))
        i_loss = (diff.abs() ** 2 + EPSILON) ** Q
        i_loss = i_loss * (1 - occmap)
        i_loss = i_loss.mean()

        flow_loss += i_weight * i_loss

    return flow_loss
```

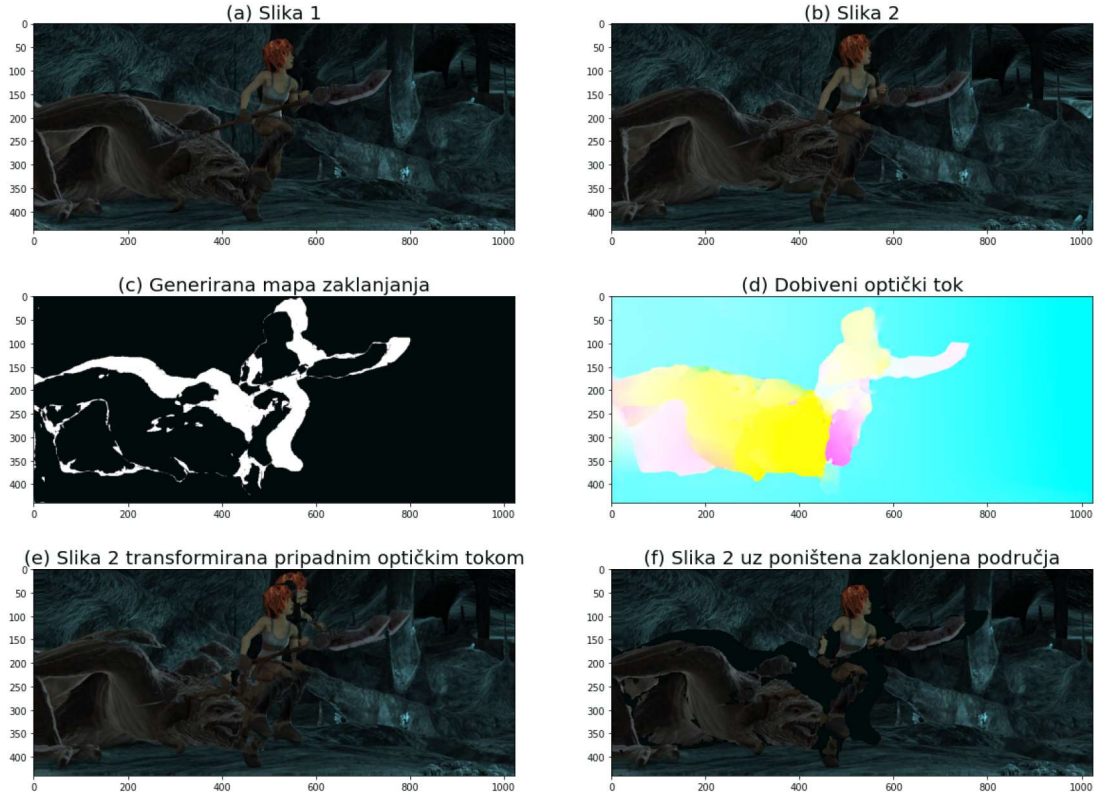
Slika 5.11: Implementacija funkcije sekvencijskog fotogubitka u programskom jeziku Python. Funkcija `flow_warp` vraća sliku 1 izobličenu korištenjem unaprijednog toka prosljeđenog kao argument, a u kombinaciju sa povratnim tokom vraća i mapu zaklanjanja generiranu prema (5.6) i (5.7).

```python
# forward pass
flow_for = model(image1, image2, iters=args.iters)
flow_bac = model(image2, image1, iters=args.iters)

# loss function
loss = sequence_photoloss(flow_for, flow_bac, image1, image2, args.gamma)

# backward pass
scaler.scale(loss).backward()
scaler.unscale_(optimizer)
torch.nn.utils.clip_grad_norm_(model.parameters(), args.clip)

# optimizing model parameters
scaler.step(optimizer)
scheduler.step()
scaler.update()
```

Slika 5.12: Struktura poziva metoda za unaprijedni prolaz, računanje funkcije gubitka, unatražni prolaz i optimizaciju parametara modela.

5.4.5. Učenje OCC modela

NOC model naučen opisanim postupkom pokazuje dobre rezultate u procjeni optičkog toka u područjima slike koja nisu zaklonjena. Očekivano, pojavljuju se problemi u procjeni toka u zaklonjenim pikselima gdje se od modela očekuje razumijevanje konteksta cijele slike i procjena zaklonjenog toka na temelju okolnih vidljivih dijelova. Opisanim postupkom učenja model nije mogao usvojiti takvo ponašanje te je potrebno dodati mehanizam učenja toka u zaklonjenim pikselima. Ograničenje nenadziranog postupka je nedostatak oznaka i zbog toga ni na koji način ne možemo doći do ispravnih oznaka u zaklonjenim dijelovima, već zaklanjanja simuliramo i pokušavamo model naučiti snalaziti se u takvim situacijama. Iz tog razloga opisani NOC model nazivamo modelom učiteljem. Učitelj je sposoban procjeniti tok u vidljvim pikselima pa njegovu procjenu koristimo kao točan tok kojim ćemo učiti model učenika nakon



Slika 5.13: Prikaz ulaznog para slika (a i b) te odgovarajućeg procijenjenog optičkog toka (d) i mape zaklanjanja (c). Slika (e) prikazuje izobličenu sliku 2 (b) korištenjem toka (d). Za potrebe računanja fotometrijskog gubitka poništavamo zaklonjena područja slike (e) na temelju mape zaklanjanja (c). Rezultat poništavanja zaklonjenih djelova je prikazan na slici (f).

što smo umjetno zaklonili vidljive piksele.

Umjetno zaklanjanje vršimo u nasumično odabranim područjima slike tako što vrijednosti odabranih piksela postavljamo na nasumične tj. područja ispunjavamo nasumičnim šumom. Odabrana područja mogu biti proizvoljne veličine i oblika, a najjednostavnija izvedba bi podrazumijevala odabir nasumičnih pravokutnih područja. Međutim, autori SelFlow-a [28] predlažu usklađivanje umjetnog zaklanjanja sa strukturom slike kako bi što vjernije simulirali zaklanjanja u stvarnim primjerima, koja naravno nisu uvijek pravilnog pravokutnog oblika. Taj cilj ostvarujemo primjenom funkcije `slic` iz paketa `skimage.segmentation`. Funkcija `slic` provodi segmentaciju slike na odabrani broj segmenata korištenjem algoritma K-središta (engl. K-means clustering) u prostoru boja i dimenzija slike. Slika 5.14 prikazuje implementaciju postupka dodavanja umjetnih zaklanjanja u drugu sliku iz ulaznog para.

Slike 5.15 i 5.16 prikazuju rezultat nasumičnog zaklanjanja područja druge slike iz ulaznog para. Lijevi dio slike prikazuje originalnu sliku iz podatkovnog skupa, a desni

```
## adding occlusions
segments = slic(image2[0].squeeze().permute(1, 2, 0).cpu().detach().numpy(), n_segments=300, compactness=800)
mask = rand.sample(list(np.unique(segments)), k=rand.randint(8, 12))

image2_orig = image2.clone().detach()
for b in image2:
    for i in range(b[0].shape[0]):
        for j in range(b[0].shape[1]):
            if segments[i][j] in mask:
                b[0][i][j] = float(rand.randrange(255))
                b[1][i][j] = float(rand.randrange(255))
                b[2][i][j] = float(rand.randrange(255))
```

Slika 5.14: Poziv funkcije `slic` za segmentiranje slike u odabrani broj segmenata te ispunjavanje nekih nasumično izabranih segmenata slučajnim šumom. Vanjska petlja iterira po svim slikama `b` iz grupe (engl. batch), a zatim iteriramo po svim pikselima slike i unosimo slučajan šum ako se piksel nalazi u nekom od nasumično odabranih segmenata.

dio istu sliku nakon dodavanja nasumičnog šuma.

Slika 5.15: Primjer dodavanja umjetnog zaklanjanja na slučajno odabrane segmente slike. Lijevi dio prikazuje originalnu sliku iz podatkovnog skupa, a desni dio istu sliku nakon dodavanja umjetnog šuma.

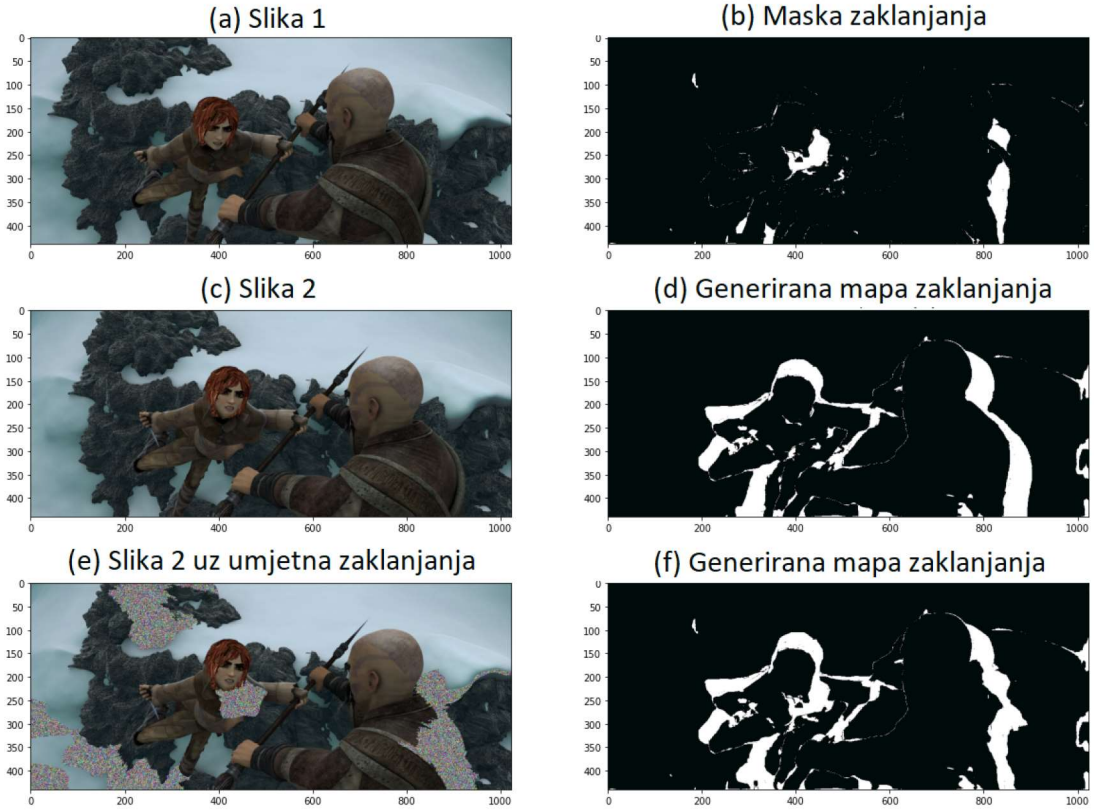
Slika 5.16: Primjer dodavanja umjetnog zaklanjanja na slučajno odabrane segmente slike. Lijevi dio prikazuje originalnu sliku iz podatkovnog skupa, a desni dio istu sliku nakon dodavanja umjetnog šuma.

Sliku na koju smo dodali zaklanjanja označujemo s $\widetilde{I}_2$. Dovođenjem para I_1, $\widetilde{I}_2$ na ulaz modela, generiramo tok $\widetilde{\mathbf{f}}_{1\to 2}$ koji osim stvarno zaklonjenih piksela na sceni, posjeduje i umjetno zaklonjene dijelove te model nije u mogućnosti ispravno procijeniti tok ni u jednom od ta dva slučaja. Međutim, za umjetno zaklonjena područja posjedujemo ispravnu oznaku toka $\mathbf{f}_{1\to 2}$ koju nam daje NOC model prije dodanih zaklanjanja. Tok $\mathbf{f}_{1\to 2}$ je generiran na paru slika I_1, I_2 i upravo tu informaciju koristimo za

učenje OCC modela. Zbog toga govorimo o modelu učitelju (NOC) i učeniku (OCC). Za izdvajanje umjetno zaklonjenih područja koristimo masku zaklanjanja $M_{1\to 2}$. Maska je u suštini razlika između mapa zaklanjanja prije i nakon dodanih zaklanjanja, a računamo je po izrazu:

$$M_{i\to j} = clip(\widetilde{O}_{i\to j} - O_{i\to j}, 0, 1) \tag{5.9}$$

Primjer generirane maske zaklanjanja $M_{1\to 2}$ prikazujemo na slici 5.17.



Slika 5.17: Primjer generiranja maske zaklanjanja $M_{1\to 2}$ (b) za izdvajanje umjetno zaklonjenih područja. NOC model na temelju ulaznog para (a i c) generira mapu zaklanjanja $O_{1\to 2}$ (d). Uz dodavanje umjetnih zaklanjanja (e), ulazni par OCC modela čine (a) i (e). OCC model generira mapu zaklanjanja $\widetilde{O}_{i\to j}$ prikazanu u (f). Mapa zaklanjanja (f) sadrži više zaklonjenih piksela (bijelih područja) nego mapa (d) zbog toga što smo dodatno, umjetno, zaklonili neke piksele. Razlika (f) i (d) je maska zaklanjanja (b) koju smo izračunali prema (5.9).

Dakle, OCC model od svog učitelja dobiva oznake za područja određena maskom $M_{1\to 2}$ i upravo te oznake koristimo u fazi učenja OCC modela. Nastojimo minimizirati razliku između dobivenih oznaka i generiranog toka kako bi model naučili snalaziti se u zaklonjenim dijelovima, a nadamo se da će to znanje model koristiti za dobru

generalizaciju u stvarnim zaklanjanjima.

Uvodimo funkciju gubitka koja kažnjava razliku toka dobivenog od učitelja $\mathbf{f}_{1\to2}$ i toka generiranog modelom učenika $\widetilde{\mathbf{f}}_{1\to2}$. Funkcija mjeri L1 normu razlike navedenih tokova, ali samo na područjima određenim maskom zaklanjanja $M_{1\to2}$.

$$L_{ss} = \sum_{i=1}^{N} \gamma^{N-i} \frac{\sum \psi(\mathbf{f}^i_{1\to2} - \widetilde{\mathbf{f}}^i_{1\to2}) \odot M_{1\to2}}{\sum M_{1\to2}} \tag{5.10}$$

Pri tome je $\psi(x) = (\mid x \mid^2 + \epsilon)^q$ funkcija charbonnierovog gubitka uz $q = 0.5$, $\epsilon = 0.001$ i $\gamma = 0.8$, a operator $\odot$ predstavlja množenje matrica po elementima. Gubitak L_{ss} možemo nazvati samonadzirani gubitak (engl. self-supervision loss).

Učenje OCC modela provodimo kombinacijom samonadziranog gubitka 5.10 i fotometrijskog gubitka slijeda 5.8.

$$L = L_{sp} + L_{ss} \tag{5.11}$$

Slika 5.18 prikazuje implementaciju navedene funkcije gubitka u programskom jeziku Python, a slika 5.19 strukturu poziva metoda prilikom učenja OCC modela.

```
def selfSup_loss(flow_preds_for, flow_preds_bac, flow_preds_for_OCC, flow_preds_bac_OCC, image1, image2_orig, gamma=0.8):
    """ Loss function defined over sequence of flow predictions """

    n_predictions = len(flow_preds_for)
    flow_loss = 0.0

    for i in range(n_predictions):
        i_weight = gamma**(n_predictions - i - 1)
        Loss = []

        warpedimg = warp_utils.flow_warp(image2_orig, flow_preds_for[i])
        occmap = utils.get_occlusion(flow_preds_for[i], flow_preds_bac[i])
        occmap_xy = torch.cat((occmap, occmap), 1)

        occmapTILDA = utils.get_occlusion(flow_preds_for_OCC[i], flow_preds_bac_OCC[i])
        occmapTILDA_xy = torch.cat((occmapTILDA, occmapTILDA), 1)

        OCCmask = torch.clamp(occmapTILDA_xy - occmap_xy, 0, 1)

        # Lss loss
        Loss += [torch.abs(flow_preds_for_OCC[i] - flow_preds_for[i].detach()) * OCCmask]

        # Lsp loss
        diff = utils.hamming_distance(utils.ternary_transform(image1), utils.ternary_transform(warpedimg))
        Loss += [diff.abs() * (1 - occmap)]

        Loss = [(l ** 2 + EPSILON) ** Q for l in Loss]
        i_loss = sum([l.mean() for l in Loss])
        flow_loss += i_weight * i_loss

    return flow_loss
```

Slika 5.18: Implementacija funkcije gubitka iz 5.11 kao kombinacija fotometrijskog gubitka slijeda i samonadziranog gubitka u programskom jeziku Python uz generiranje potrebnih mapa zaklanjanja $O_{1\to2}$ i $\widetilde{O}_{1\to2}$ i maske zaklanjanja $M_{i\to j}$.

5.5. Prijedlog redoslijeda učenja i korištene postavke

Slijedeći upute iz literature [43], provodimo pred-učenje (engl. pretraining) na skupovima Flying Chairs te zatim Flying Things. Nakon toga model ugađamo (engl. fine-tune) na specifičnim skupovima kao što su Sintel i Cityscapes. Budući da postupkom

```
# NOC model forward pass
flow_for = model(image1, image2, iters=args.iters)
flow_bac = model(image2, image1, iters=args.iters)

## adding occlusions
segments = slic(image2[0].squeeze().permute(1, 2, 0).cpu().detach().numpy(), n_segments=300, compactness=800)
mask = rand.sample(list(np.unique(segments)), k=rand.randint(8, 12))

image2_orig = image2.clone().detach()
for b in image2:
    for i in range(b[0].shape[0]):
        for j in range(b[0].shape[1]):
            if segments[i][j] in mask:
                b[0][i][j] = float(rand.randrange(255))
                b[1][i][j] = float(rand.randrange(255))
                b[2][i][j] = float(rand.randrange(255))


# OCC model forward pass
flow_for_OCC = model(image1, image2, iters=args.iters)
flow_bac_OCC = model(image2, image1, iters=args.iters)

# loss function L = Lsp + Lss
loss = sequence_OCCloss(flow_for, flow_bac, flow_for_OCC, flow_bac_OCC, image1, image2_orig, args.gamma)

if torch.isnan(loss) or torch.isinf(loss):
    del loss
    continue

# backward pass
scaler.scale(loss).backward()
scaler.unscale_(optimizer)
torch.nn.utils.clip_grad_norm_(model.parameters(), args.clip)

# parameter optimizing
scaler.step(optimizer)
scheduler.step()
scaler.update()
```

Slika 5.19: Struktura poziva metoda za unaprijedni prolaz, računanje funkcije gubitka, unatražni prolaz i optimizaciju parametara modela. Treba primjetiti da su NOC i OCC model predstavljeni istom instancom objekta. Željeno ponašanje učitelja i učenika postižemo višestrukim unaprijednim prolazima i dodavanjem umjetnih zaklanjanja. Na taj način istovremeno učimo učitelja i učenika. Drugi pristup bi u prvoj fazi najprije generirao oznake učitelja koje bi učenik koristio u sljedećoj fazi što bi bilo računalno manje zahtjevno, ali tada ne bismo kontinuirano unapređivali učitelja kao što je to slučaj ovdje. U vrlo složenim slikama, a pogotovo nakon provedbe transformacija na slikama, modeli koji se nalaze tek u početnoj fazi učenja mogu pogrešno prepoznati cijelu sliku kao zaklonjenu. Zbog takvih slučajeva dodajemo provjeru iznosa izračunatog gubitka kako bi izbjegli eventualna pojavljivanja neispravnih vrijednosti.

opisanim u 5.4.5 simultano unapređujemo učitelja i učenika, ta završna ugađanja modela nije potrebno provoditi u opisane dvije faze, već možemo odmah učiti funkcijom (5.11). Pred-učenjem je model učitelja dovoljno napredovao da može generirati ispravne oznake u nezaklonjenim dijelovima pa nije potrebno ponovno učiti NOC model tijekom finog ugađanja, a uz to, učitelj će napredovati i tijekom učenja OCC modela. NOC model učimo samo na početku kada su parametri modela potpuno nasumično inicijalizirani pa je učitelj previše loš da bi navodio učenika. Tablica 5.2 prikazuje prijedlog redoslijeda učenja modela s obzirom na podatkovne skupove. Treba primjetiti da NOC model učimo samo u prvom koraku, a nakon toga je učitelj dovoljno napredovao da može navoditi učenje učenika pa na svakom sljedećem skupu odmah učimo OCC model.

Kompletan postupak učenja provodimo na Colab platformi. Colaboratory ili skra-

Redni broj	Podatkovni skup	Redni broj faze iz koje učitavamo naučene težine	Funkcija gubitka	Stopa učenja	Veličina grupe za učenje (engl. batch size)	Broj iteracija
1.	Flying Chairs	- (nasumična inicijalizacija)	L_{sp} (5.8)	1e-4	4	200 000
2.	Flying Chairs	1.	$L = L_{sp} + L_{ss}$ (5.11)	1e-5	2	100 000
3.	Flying Things	2.	$L = L_{sp} + L_{ss}$ (5.11)	1e-5	2	50 000
4.	Sintel	3.	$L = L_{sp} + L_{ss}$ (5.11)	1e-5	2	50 000
	Cityscapes	3.	$L = L_{sp} + L_{ss}$ (5.11)	1e-5	2	50 000

Tablica 5.2: Prijedlog redoslijeda učenja

ćeno Colab je javno dostupna platforma za pisanje i izvršavanje koda u programskom jeziku Python. Platforma je proizvod tvrtke Google i namijenjena je svima pa se može koristiti u istraživačke i akademske svrhe. Korištenje ne zahtjeva instalaciju bilo kakve programske podrške, već je pristup platformi ostvaren putem bilo kojeg web preglednika. U pozadini, Colab korisniku omogućuje pristup GPU jedinicama te pruža određenu količinu memorije za privremenu pohranu. Poveznica između korisnikovog programskog koda i colab resursa je usluga Jupyter bilježnice koja se učitava u korisnikov web preglednik. Resursi koje Colab nudi su ograničeni i ovise o planu pretplate korisnika. Osim navedenog, Colab nudi mogućnost povezivanja radne okoline s Google Drive uslugom za pohranu podataka te na taj način korisnik ima mogućnost učitavanja potrebnih datoteka kao što su npr. podatkovni skupovi.

Prije početka učenja, težine modela postavljamo na nasumične vrijednosti. Nakon toga, težine ažuriramo AdamW [29] optimizatorom uz obrezivanje gradijenata na interval $[-1, 1]$. Stopa učenja u pojedinoj fazi je navedena u tablici 5.2, a tijekom učenja koristimo OneCycle plan kretanja stope učenja. Colab nam omogućuje korištenje samo jedne GPU jedinice za učenje, a to ograničenje nam onemogućuje isprobavanje nekih drugih postavki jednako kao i ograničenje u dostupnoj GPU memoriji koje nas ograničava na veličinu grupe za učenje (engl. batch size) na najviše 2 tijekom učenja OCC modela. Povećanje dostupnih resursa je zasigurno velika mogućnost napretka u ovom radu.

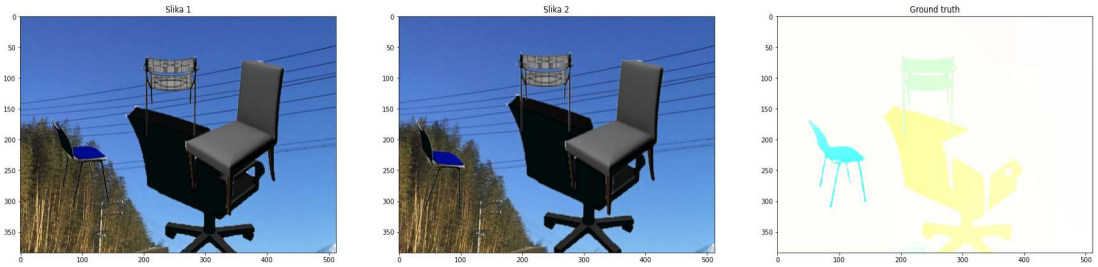
Budući da je RAFT iterativan model, broj iteracija ugađanja toka tijekom faze učenja je također jedan od hiperparametara postupka. Tijekom faze učenja, koristimo 12 iteracija ugađanja toka.

6. Skupovi podataka

U ovom poglavlju ćemo navesti podatkovne skupove korištene za samonadzirano učenje i evaluaciju u ovom radu. Skupovi Flying Chairs, Flying Things i Sintel posjeduju oznake točnih optičkih tokova za svaki par slika dok skup Cityscapes nema oznaka.

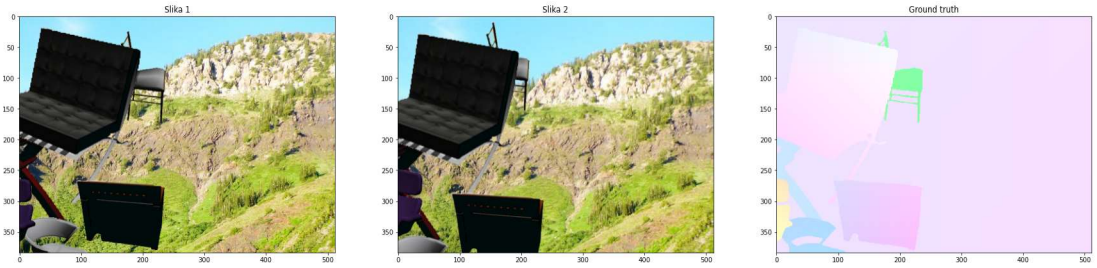
6.1. Flying Chairs

Flying Chairs je računalno generiran podatkovni skup za optički tok. Sastoji se od 22872 parova slika u *.ppm* formatu s pripadnim označenim točnim tokom. Skup je generiran korištenjem 3D modela različitih oblika stolica koje se kreću u nasumičnim smjerovima, a u pozadini su smještene nasumične slike iz baze Flickr. Gibanja stolica i pozadine su u potpunosti planarna što znači da se događaju u istoj ili paralelnim ravninama. Podatkovni skup je generiran za potrebe učenja konvolucijskog modela FlowNet u radu iz 2015. godine [11]. U trenutku objave, skup Flying Chairs je predstavljao veliki napredak budući da je sadržavao daleko veći broj podataka od tada postojećih sličnih skupova. Slike 6.1, 6.2 i 6.3 prikazuju nasumične primjere iz Flying Chairs skupa zajedno s odgovarajućim točnim tokom.



Slika 6.1: Primjer para slika iz skupa Flying Chairs te odgovarajući točni optički tok.

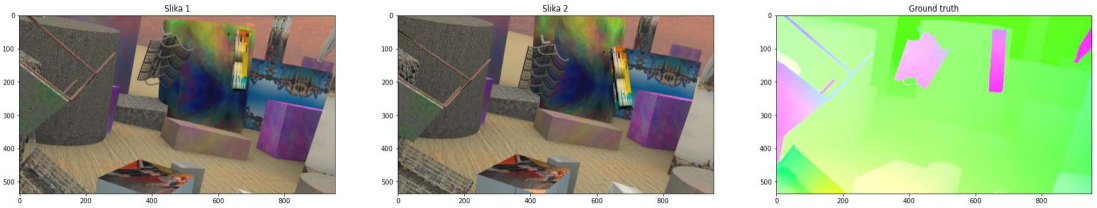
Slika 6.2: Primjer para slika iz skupa Flying Chairs te odgovarajući točni optički tok.



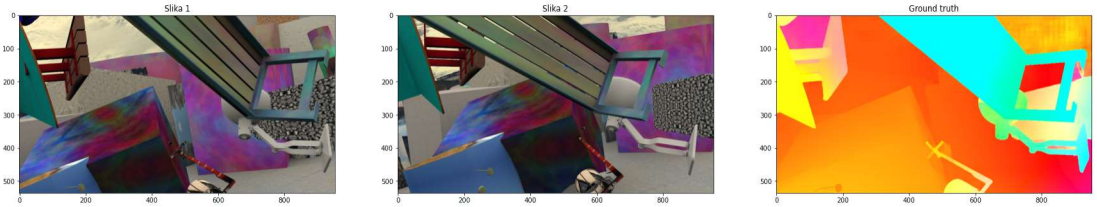
Slika 6.3: Primjer para slika iz skupa Flying Chairs te odgovarajući točni optički tok.

6.2. Flying Things

Skup Flying Things se sastoji od preko 39000 stereo slika generiranih iz različitih umjetno stvorenih uzoraka. Skup je namjenjen istraživačima te je besplatan i javno dostupan [30]. Skup sadrži odgovarajući točni optički tok za svaki par slika kao i mape dubine. Slike 6.4 i 6.5 prikazuju nasumične primjere iz skupa Flying Things.



Slika 6.4: Primjer para slika iz skupa Flying Things te odgovarajući točni optički tok.

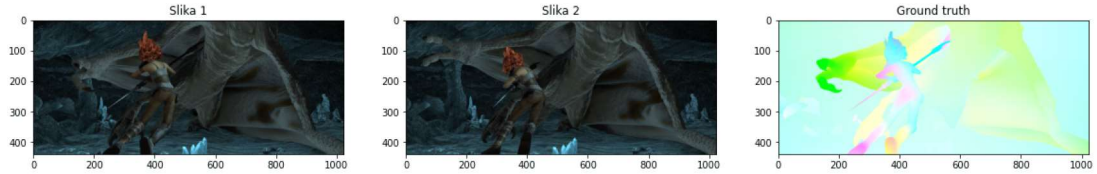


Slika 6.5: Primjer para slika iz skupa Flying Things te odgovarajući točni optički tok.

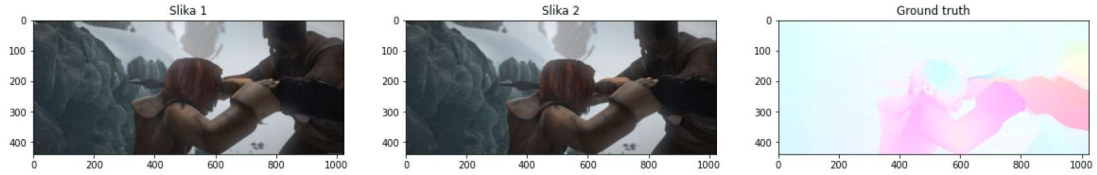
6.3. Sintel

Podatkovni skup Sintel je nastao izlučivanjem slikovnih isječaka iz istoimenog kratkog animiranog filma. Slike su potom uređene kako bi bile primjerene za korištenje u skupu za optički tok. Skup se sastoji od 1064 slike podjeljene u 23 scene. Sve slike su u RGB formatu i svaki par slika popraćen je točnim optičkim tokom, mapom zaklanjanja i dubinom scene. Skup ima brojne prednosti u odnosu na druge skupove za optički tok, neke od njih su zamućenja (engl. motion blur), zaklonjena područja koja nemaju korespondentne piksele u susjednim slikama te mogućnost procjene toka analizom više od dvije uzastopne slike. Skup se sastoji od dvije inačice svake scene: clean pass i final pass. Final pass sadrži zamućenja zbog pokreta, promjene u kontrastima te promjene u osvjetljenjima i sjene koje otežavaju procjenu optičkog toka. Slike 6.6, 6.7 i 6.8 prikazuju nasumične primjere iz skupa Sintel clean te odgovarajući točni optički tok.

Slika 6.6: Primjer para slika iz skupa Sintel clean te odgovarajući točni optički tok.



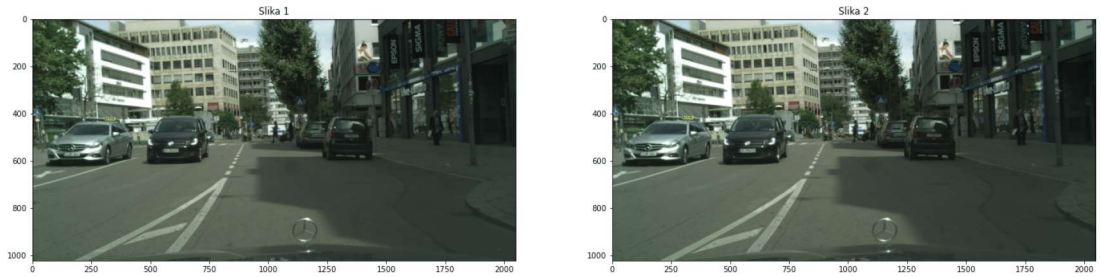
Slika 6.7: Primjer para slika iz skupa Sintel clean te odgovarajući točni optički tok.



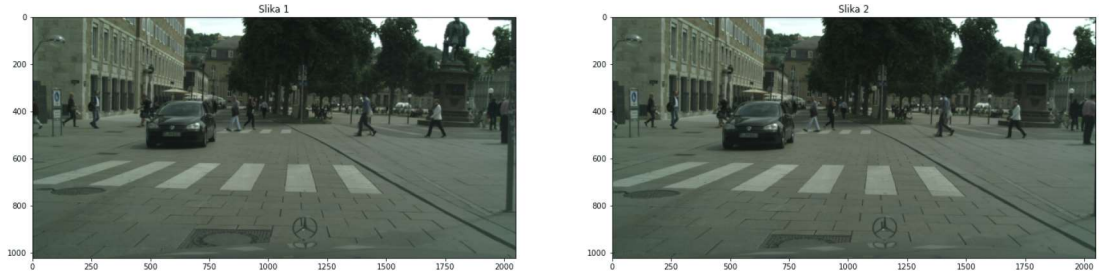
Slika 6.8: Primjer para slika iz skupa Sintel final te odgovarajući točni optički tok.

6.4. Cityscapes

Podatkovi skup Cityscapes sadrži stereoskopske slike snimljene na prometnicama 50 različitih gradova. Slike sadrže oznake segmenata objekata kao i klasa pojedinih segmenata na slikama, no skup ne sadrži točan optički tok. Skup je namjenjen modelima za semantičku segmentaciju slika, a budući da sadrži nizove uzastopnih slika, može se koristiti za vizualnu usporedbu uspješnosti različitih modela za optički tok. Slike 6.9 i 6.10 prikazuju nasumične primjere iz skupa Cityscapes.



Slika 6.9: Primjer para uzastopnih slika iz skupa Cityscapes.



Slika 6.10: Primjer para uzastopnih slika iz skupa Cityscapes.

7. Eksperimenti i rezultati

Ovo poglavlje analizira i vrednuje modele koje smo naučili predloženim postupkom. Naš cilj bio je usporediti predloženi postupak učenja s ostalim nenadziranim modelima te pokušati reproducirati ili čak nadmašiti rezultate najboljih poznatih nenadziranih modela, ali i usporediti te rezultate sa nadziranim postupcima. Najprije će biti opisane standardne mjere kojima evaluiramo uspješnost modela, a zatim ćemo dobivene modele evaluirati i usporediti s drugima. Procijenjeni optički tok ćemo vizualizirati te prikazati primjere procjene toka na u slikama iz korištenih podatkovnih skupova.

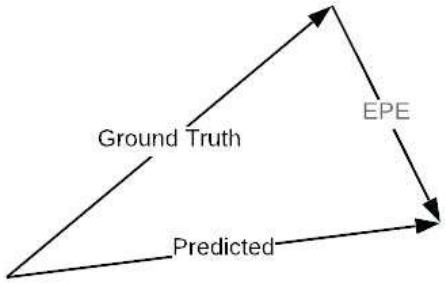
7.1. Metrike evaluacije

Mjera uspješnosti procjene optičkog toka koja je najšire prihvaćena među autorima je EPE mjera pogreške (engl. end-to-end point error). EPE mjera predstavlja euklidsku udaljenost procijenjenog i točnog (engl. ground truth) vektora optičkog toka za pojedini piksel. Geometrijska reprezentacija pogreške je prikazana na slici 7.1. EPE mjera za pojedini piksel se računa prema (7.1). Međutim, uspješnost evaluacije promatramo na cijelom području ulaznih slika pa zbog toga koristimo prosječnu EPE mjeru po svim pikselima slike, odnosno u slučaju evaluacije na podatkovnom skupu, prosječnu EPE mjeru po svim pikselima iz svih slika u skupu. Slika 7.2 prikazuje rezultat evaluiranja dobivenog toka EPE mjerom.

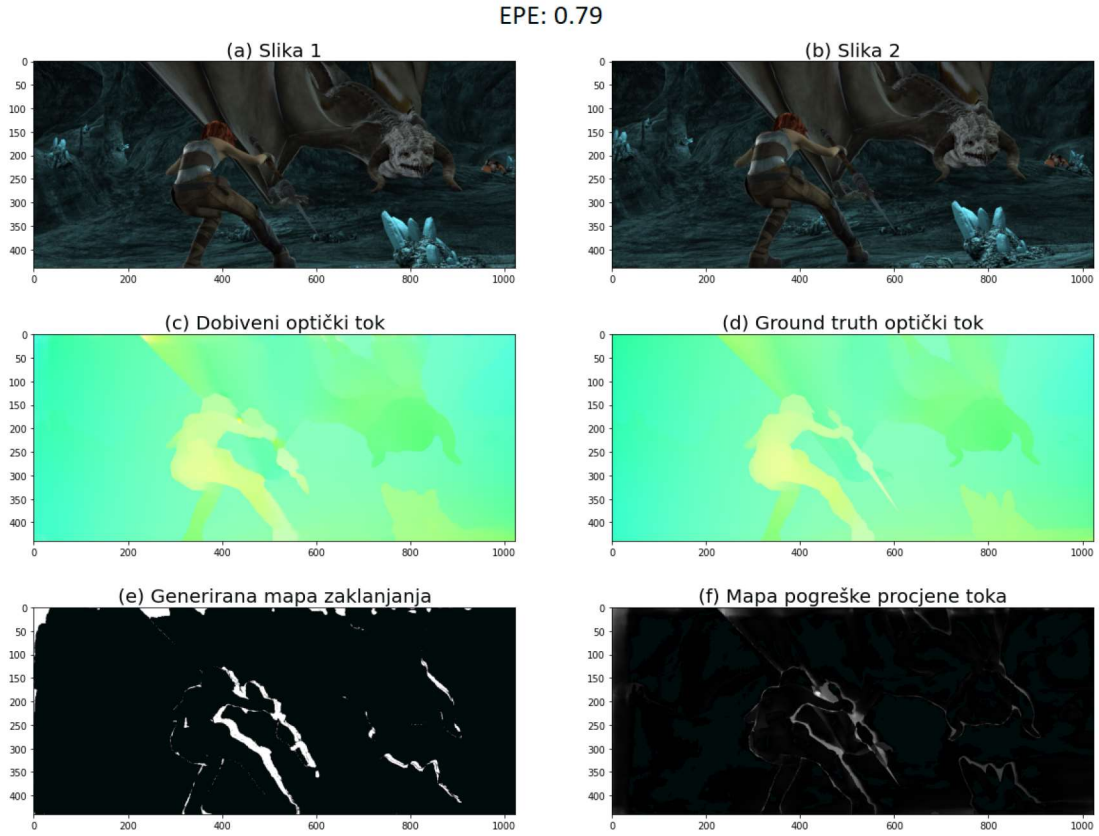
$$EPE = \sqrt{(\mathbf{f}_x - \mathbf{f}_x^{GT})^2 + (\mathbf{f}_y - \mathbf{f}_y^{GT})^2} \tag{7.1}$$

Gdje je $\mathbf{f}^{GT}$ označeni, a $\mathbf{f}_x$ procjenjeni optički tok.

Osim EPE mjere, česta mjera koja se koristi u literaturi je F1 mjera koja mjeri udio ispravno procjenjenih vektora toka među svim pikselima. F1 mjera se češće koristi u radu s rijetkim tokom (engl. sparse flow) budući da u tom slučaju ne postoje oznake za sve piksele nego samo za određeni podskup. Tada F1 mjera govori za koliki postotak piksela je tok ispravno procijenjen. Dakle, vrijednosti F1 mjere su u intervalu $[0, 1]$.



Slika 7.1: EPE mjera pogreške procjene optičkog toka. **Izvor:** https://medium.com/swlh/what-is-optical-flow-and-why-does-it-matter-in-deep-learning-b3278bb205b5



Slika 7.2: Primjer izračuna EPE mjere pogreške toka na temelju procijenjenog toka (c) i dostupnih oznaka (d). Mapa pogreške (f) prikazuje EPE pogrešku u svakom pojedinom pikselu na način da veći intenzitet svjetline piksela predstavlja veću pogrešku, a ukupna EPE mjera se računa kao prosjek vrijednosti iz (f) po svim pikselima.

7.2. Utjecaj pojedinih parametara na kvalitetu toka

Za početak ćemo analizirati utjecaj hiperparametra postupka učenja. Odjeljak 5.2 opisuje transformacije ulaznih podataka koje koristimo za učenje. Jedna od stvari koje

je zanimljivo testirati je utjecaj asimetrične fotometrijske transformacije ulaznog para. U eksperimentu postavljamo vjerojatnost asimetrične transformacije tijekom učenja $p_{asym} = 0.2$ što znači da će u prosjeku u svakom petom paru slika 1 biti transformirana drugačijim postavkama od slike 2. Tablica 7.1 uspoređuje dobivene rezultate na skupovima Flying Chairs i Sintel nakon treniranja uz $p_{asym} = 0.2$ i $p_{asym} = 0$.

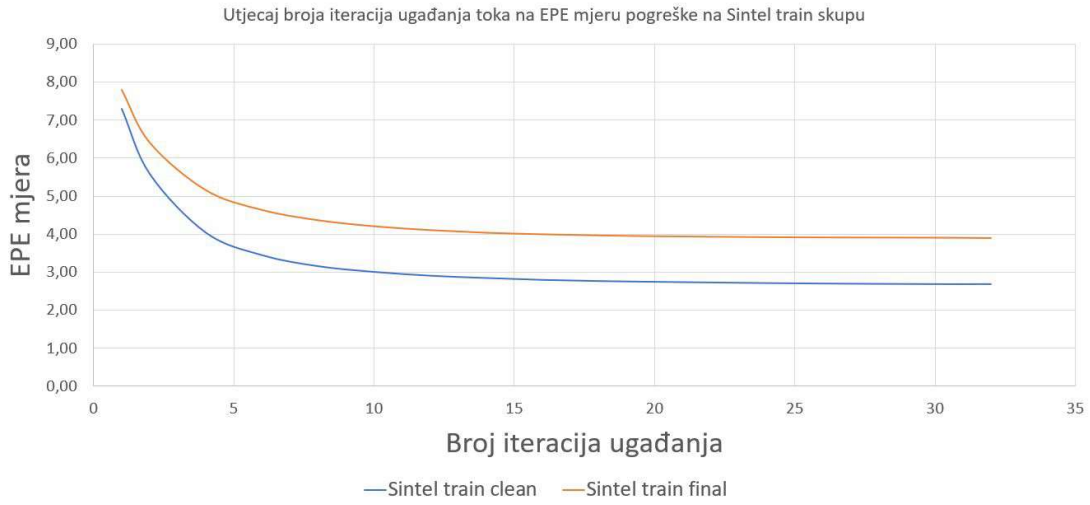
Skup za učenje	Model	Redoslijed učenja i korištene postavke	Najbolji rezultati dobiveni evaluacijom EPE mjera		
			Flying Chairs (test)	Sintel train clean pass	Sintel train final pass
Flying Chairs (train)	RAFT (5M params)	NOC + OCC, $p_{asym} = 0$	**2.90**	**3.06**	**4.12**
		NOC + OCC, $p_{asym} = 0.2$	3.05	3.36	4.32

Tablica 7.1: Utjecaj parametra vjerojatnosti asimetrične transformacije para slika tijekom učenja na točnost procijenjenog toka.

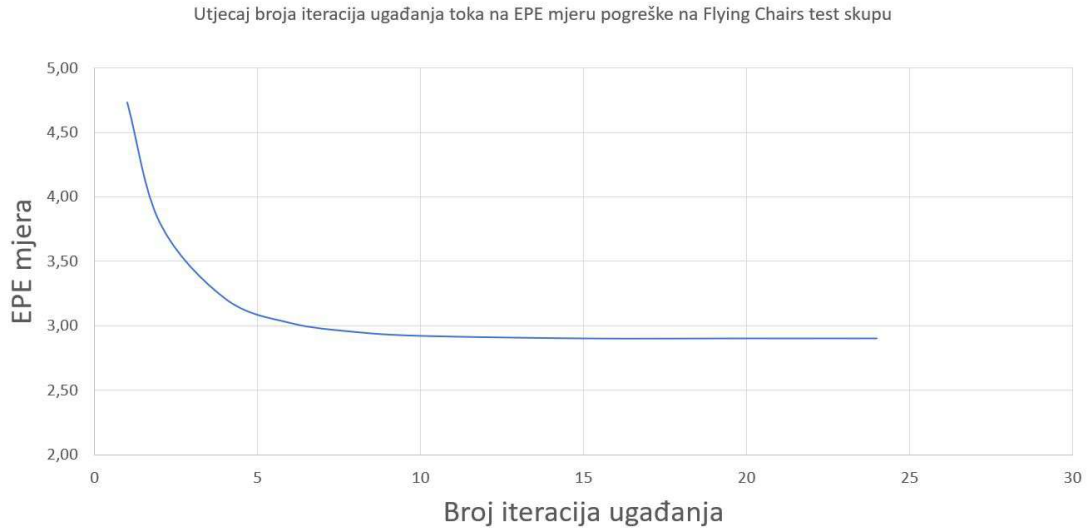
Možemo zaključiti da asimetrična fotometrijska transformacija ulaznog para ne doprinosi uspješnosti procjene toka. To možemo objasniti činjenicom da učenje obavljamo fotometrijskim gubitkom koji kažnjava razliku u intenzitetu piksela izvorne i izobličene slike. Ukoliko par slika transformiramo na različite načine, intenziteti istog piksela u dvjema slikama neće biti isti pa ih postupak ne može ispravno upariti. Asimetrična transformacija ima smisla kod nadziranog postupka [43] gdje posjedujemo oznaku toka, a slike transformiramo asimetrično kako bismo postigli bolju generalizaciju, no u nenadziranom postupku taj trik ne možemo koristiti.

Budući da je RAFT iterativan model, možemo promatrati utjecaj provedenih iteracija ugađanja na kvalitetu dobivenog toka.

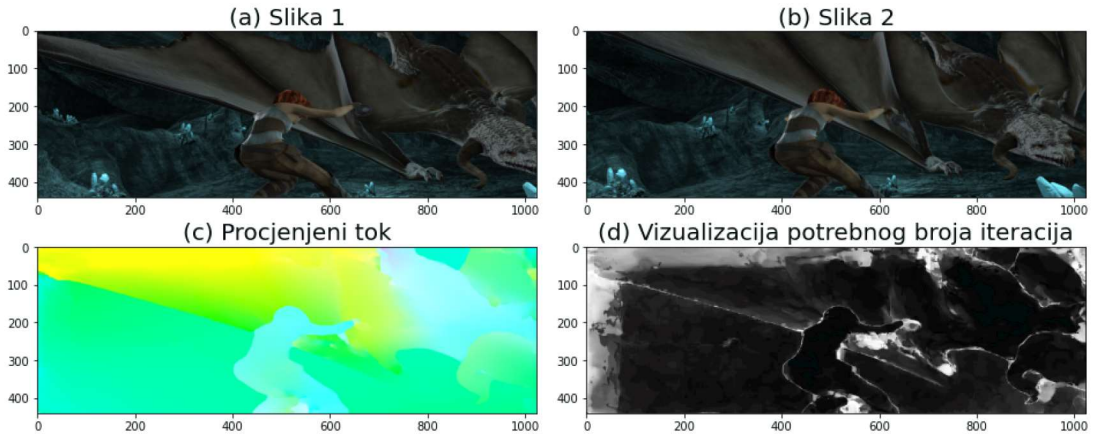
Slike 7.3 i 7.4 pokazuju da funkcija prosječne EPE pogreške toka, koja ovisi o broju iteracija ugađanja, relativno brzo konvergira. To znači da vrijednosti ugađanja toka tijekom iteracija teže k nuli, a ukupan tok generiran modelom teži fiksnoj vrijednosti $\mathbf{f}^*$. Upravo to je važno svojstvo RAFT modela budući da imamo mogućnost provedbe proizvoljnog broja iteracija bez opasnosti od divergencije u procjeni toka. Slike 7.5 i 7.6 prikazuju broj potrebnih iteracija ugađanja toka za svaki pojedini piksel na primjeru ulaznog para slika iz skupa Sintel odnosno Flying Chairs, a slika 7.7 pokazuje napredovanje postupka tijekom iteracija na paru slika iz skupa Sintel.



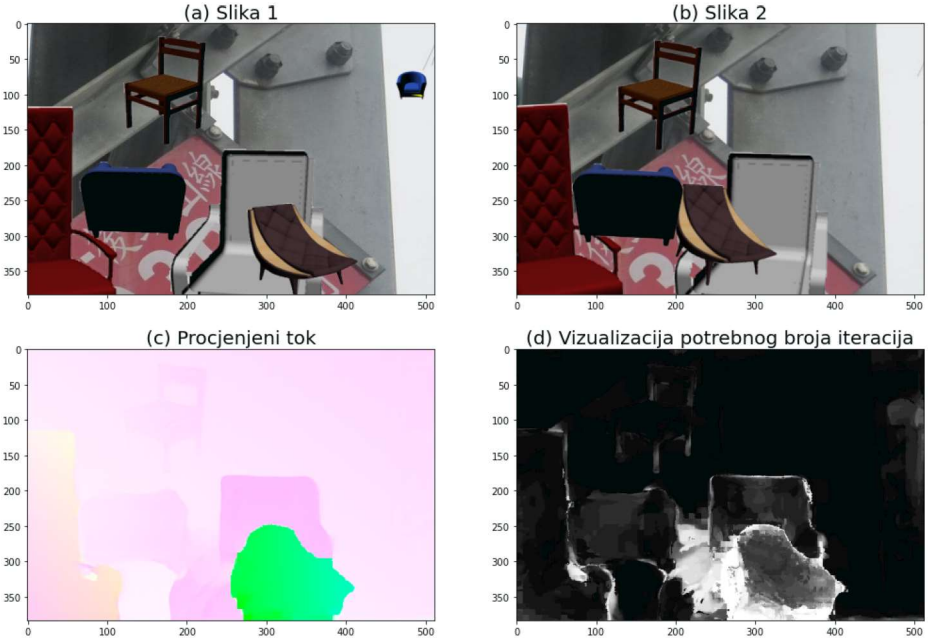
Slika 7.3: Utjecaj broja provedenih iteracija ugađanja na kvalitetu procjene optičkog toka na podatkovnom skupu Sintel train.



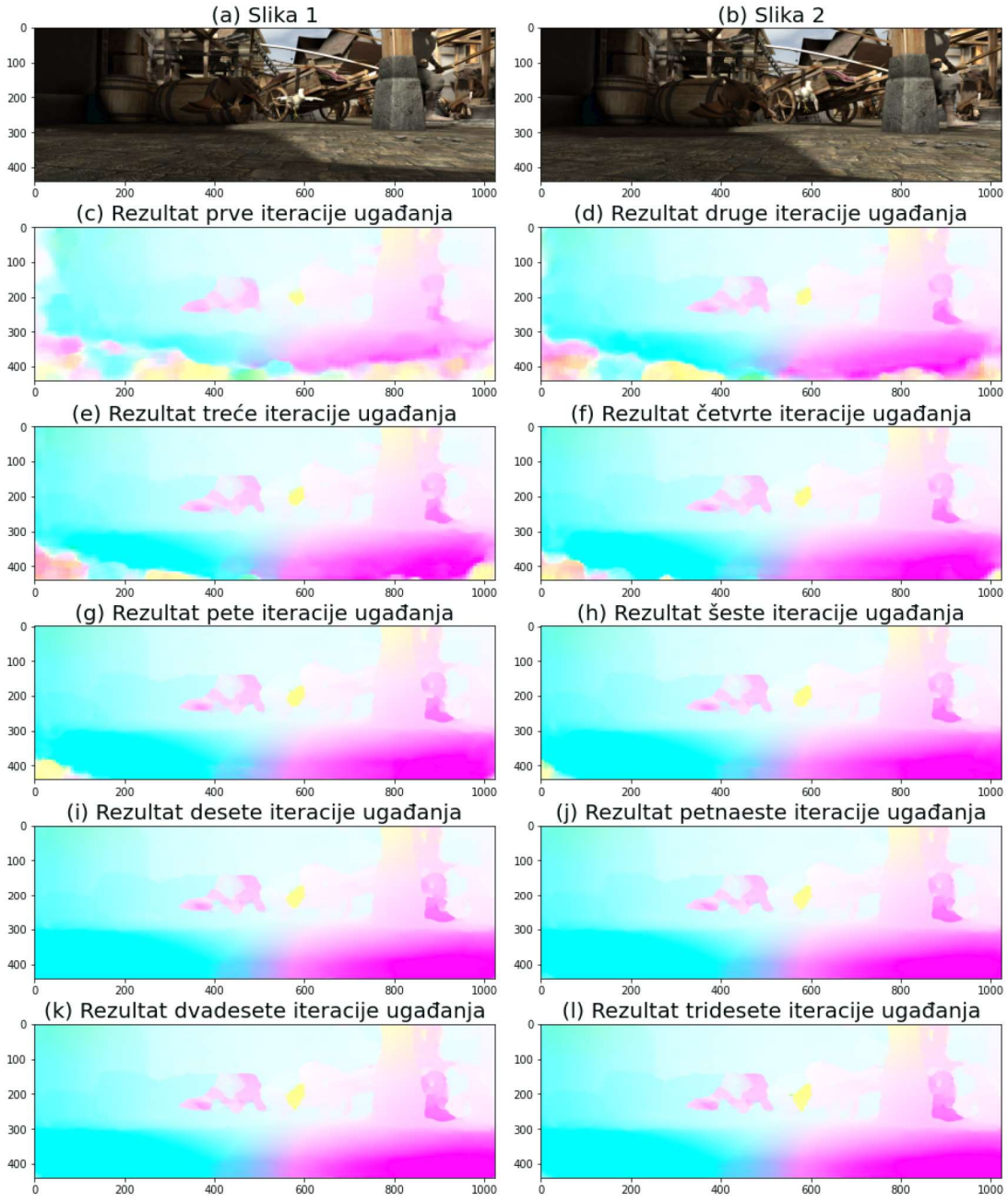
Slika 7.4: Utjecaj broja provedenih iteracija ugađanja na kvalitetu procjene optičkog toka na podatkovnom skupu Flying Chairs test.



Slika 7.5: Vizualizacija potrebnog broja iteracija ugađanja u pojedinim pikselima. Slike (a) i (b) prikazuju ulazni par slika iz skupa Sintel train, a (c) procijenu toka dobivenu našim modelom. Slika (d) opisuje potreban broj iteracija ugađanja za svaki pojedini piksel kako bi se došlo do konačne procijene toka iz (c). Veći intenzitet piksela označava veći broj potrebnih iteracija da bi se došlo do konačne procijene uz toleranciju od jednog piksela. Očekivano, najveći broj iteracija je potreban u područjima gdje postoje zaklanjanja objekata.



Slika 7.6: Vizualizacija potrebnog broja iteracija ugađanja u pojedinim pikselima. Slike (a) i (b) prikazuju ulazni par slika iz skupa Flying Chairs, a (c) procijenu toka dobivenu našim modelom. Slika (d) opisuje potreban broj iteracija ugađanja za svaki pojedini piksel kako bi se došlo do konačne procijene toka iz (c). Veći intenzitet piksela označava veći broj potrebnih iteracija da bi se došlo do konačne procijene uz toleranciju od jednog piksela. Očekivano, najveći broj iteracija je potreban u područjima gdje postoje zaklanjanja objekata.



Slika 7.7: Prikaz napredovanja postupka procijene optičkog toka tijekom iteracija ugađanja. Slike (a) i (b) prikazuju ulazni par slika, a slike (c)-(l) procjene optičkog toka našeg modela tijekom uzastopnih iteracija ugađanja.

U literaturi se često kao najvažniji mjerodavni podatkovni skup za evaluaciju modela za optički tok koristi podatkovni skup Sintel. Iz toga razloga ćemo analizirati utjecaj učenja na različitim podatkovnim skupovima na kvalitetu toka na podatkovnom skupu Sintel train. Rezultate pojedinih faza učenja na podatkovnim skupovima Flying Chairs i Flying Things prikazujemo u tablici 7.2 uz korištenje obje implementacije modela: RAFT i RAFT-S. Tablicom 7.3 prikazujemo utjecaj samonadziranog ugađanja na skupu Sintel train clean. Tablica 7.4 prikazuje vremena trajanja unaprijednog prolaza za jedan par slika te trajanje pojdinih faza učenja.

Oznaka faze učenja	Model	Skup za učenje	Oznaka faze učenja iz koje učitavamo težine + model učenja	Najbolji rezultati dobiveni evaluacijom EPE mjera	
				Sintel train clean pass	Sintel train final pass
a	RAFT (5M params)	Flying Chairs (train)	Nasumična inicijalizacija (-) + NOC	6.917	7.515
b		Flying Chairs (train)	(a) + OCC	3.060	4.121
c		Flying Things (train)	(b) + OCC	2.876	4.215
d	RAFT-S (1M params)	Flying Chairs (train)	Nasumična inicijalizacija (-) + NOC	5.779	6.784
e		Flying Chairs (train)	(d) + OCC	4.256	5.310
f		Flying Things (train)	(e) + OCC	3.657	4.750

Tablica 7.2: Utjecaj faza treniranja na uspješnost procjene toka na sintel train podatkovnom skupu. Niti jedna od ovih faza učenja ne uključuje skup Sintel u svoj postupak, već promatramo kako koja faza pred-učenja utječe na konačan rezultat. Samonadzirano ugađanje na skupu Sintel ćemo obaviti kasnije. Učenje je u svakoj fazi potpuno samonadzirano. U fazama (a) i (d) model inicijaliziramo nasumičnim težinama te učimo NOC model na skupu Flying Chairs prema postupku opisanom u 5.4.4, razlika je samo u broju parametara korištene implementacije. Analogno tome, u fazama (b) i (e) koristimo težine naučene u (a) odnosno (d) te nastavljamo učenje OCC modela na skupu Flying Chairs prema postupku iz 5.4.5. Faze (c) i (f) nastavljaju učenje OCC modela, ali uz korištenje skupa Flying Things. Zanimljivo je primjetiti da je učitavanje težina naučenih OCC modelom učenja na Flying Chairs skupu i nastavak učenja na Flying Things skupu poboljšalo rezultat na clean prolazu skupa Sintel, a istovremeno pogoršalo rezultat na final prolazu.

Model	Redoslijed treniranja na skupovima uz korišteni model učenja	Najbolji rezultati dobiveni evaluacijom EPE mjera	
		Sintel train clean pass	Sintel train final pass
RAFT (5M params)	Flying Chairs NOC + Flying Chairs OCC + Flying Things OCC	2.876	4.215
	Flying Chairs NOC + Flying Chairs OCC + Flying Things OCC + Sintel train clean OCC	{2.693}	3.898

Tablica 7.3: Utjecaj samonadziranog ugađanja modela na skupu Sintel train clean. Vitičasta zagrada označava da je učenje obavljeno na tom skupu na kojem evaluiramo, ali bez korištenja točnih oznaka. Drugi stupac tablice opisuje redoslijed učenja. Učenje je obavljeno slijedno prema opisu iz tablice 7.2 te treba primijetiti da rezultati u prvom retku ove tablice odgovaraju rezultatima faze (c) iz tablice 7.2 jer su to isti modeli. U drugom retku koristimo model iz spomenute faze (c) i provodimo samonadzirano ugađanje na skupu Sintel train clean bez korištenja točnih oznaka, što prikazujemo vitičastim zagradama.

Model	Trajanje procijene toka na jednom paru slika (BS = 1)		Trajanje učenja po paru slika (BS = 1)			Ukupno trajanje učenja	
	Flying Chairs	Sintel	Flying Chairs		Sintel	NOC model (200K iteracija)	OCC model (100K iteracija)
UnRAFT (5M params)	0.13 s	0.22 s	NOC model	OCC model	OCC model	∼24 h	∼48 h
			0.35 s	1.50 s	1.58 s		
UnRAFT-S (1M params)	0.05 s	0.08 s	0.17 s	1.05 s	1.20 s	∼15 h	∼36 h

Tablica 7.4: Trajanje procijene optičkog toka na pojedinačnim parovima slika te trajanje pojedinih faza učenja u ovisnosti o inačici modela. Vremena trajanja smo mjerili uz korištenje resursa dostupnih na opisanoj platformi Colab.

7.3. Usporedba rezultata s drugim modelima

Tablica 7.5 uspoređuje rezultate različitih modela za optički tok na mjerodavnim skupovima. Modeli su grupirani prema paradigmi učenja i skupovima korištenima za učenje.

			Prosječna EPE mjera na skupu					
			Sintel train		Sintel test		KITTI 2015 train	
	Korišteni skupovi za učenje	Model	clean	final	clean	final	epe	F1
Nadzirani	Flying Chairs + Flying Things	(A) FlowNet2 [20]	2.02	3.54	**3.96**	**6.02**	10.08	30.0 %
		(B) PWC-Net [41]	2.55	3.93	-	-	10.35	33.7 %
		(C) LiteFlowNet [18]	2.48	4.04	-	-	10.39	28.5 %
		(D) MaskFlowNet [45]	2.25	3.61	-	-	-	**23.1 %**
		(E) RAFT [43]	**1.43**	**2.71**	-	-	**5.04**	**17.4 %**
	Flying Chairs + Flying Things + Sintel / KITTI	(F) SpyNet+ft [35]	(3.17)	(4.32)	6.64	8.36	-	-
		(G) FlowNet2+ft [20]	(1.45)	(2.01)	4.16	5.74	(2.3)	**(6.8 %)**
		(H) LiteFlowNet+ft [18]	(1.35)	(1.78)	4.54	5.38	(1.62)	-
		(I) PWC-Net+ft [41]	(1.71)	(2.34)	3.45	4.60	(1.45)	-
		(J) SelFlow+ft [28]	(1.68)	(1.77)	3.74	4.26	(1.18)	-
		(K) RAFT [43]	**(0.77)**	**(1.20)**	**2.08**	**3.41**	**(0.64)**	**(1.5 %)**
Nenadzirani	Sintel / KITTI	(L) OAFlow [44]	{4.03}	{5.95}	7.95	9.15	{8.88}	-
		(M) UFlow-train [22]	**{2.50}**	**{3.39}**	5.21	6.50	**{2.71}**	**{9.05 %}**
		(N) SelFlow [28]	{2.88}	{3.87}	6.56	6.57	{4.84}	-
		(O) DDFlow [27]	{2.92}	{3.98}	6.18	7.40	{5.72}	-
		(P) UnrolledCost [26]	{2.75}	{3.61}	**4.69**	**5.80**	{2.87 }	-
	SYNTHIA	(Q) UnFlow [31]	-	7.91	-	10.21	8.10	23.27 %
	Flying Chairs + Flying Things	**(R) UnRAFT (5 M)**	2.88	4.22	5.02	7.42	11.61	25.3 %
		(S) UnRAFT-S (1 M)	3.65	4.75	14.50	21.40	14.07	31.5 %
	Flying Chairs + Flying Things + Sintel	**(T) UnRAFT (5 M)**	{2.69}	3.90	4.77	7.29	-	-

Tablica 7.5: Usporedba dobivenih rezultata s drugim modelima. Tablica je podijeljena na dvije glavne cjeline, nadzirani i nenadzirani modeli. Nenadzirani RAFT model opisan u ovom radu označen je s UnRAFT (R, T) odnosno UnRAFT-S (S). Podebljani rezultati predstavljaju najbolje rezultate EPE mjere na pojedinim skupovima među skupinom modela koje ima smisla uspoređivati. Rezultati u oblim zagradama govore da je rezultat dobiven na skupu koji je korišten za učenje uz korištenje točnih oznaka. Primjetimo da su pojedini nenadzirani modeli učeni na skupovima na kojima su evaluirani, ali nenadzirano što znači da nisu korištene oznake tijekom učenja. Takvi rezultati su označeni vitičastim zagradama. Možemo zaključiti da naš nenadzirani model (T) postiže rezultate vrlo usporedive s najboljim nenadziranim modelima (M, P, N). Osim toga pokazujemo da se vrlo dobri rezultati (R) mogu postići bez učenja na skupovima Sintel što čini razliku u odnosu na ostale koji su učeni i evaluirani na Sintel train.

Tablica 7.6 uspoređuje rezultate različitih modela za optički tok na Flying Chairs test skupu. Modeli su učeni na skupu Flying Chairs train, a tablica je podjeljena u dvije osnovne grupe, nadzirani i nenadzirani modeli.

	Model	Prosječna EPE mjera na skupu Flying Chairs	
		Train	Test
Nadzirani	(A) FlowNet2 [20]	-	1.78
	(B) PWC-Net [41]	-	2.00
	(C) RAFT [43]	**0.77**	**0.82**
	(D) SpyNet [35]	-	2.63
Nenadzirani	(E) UFlow-train [22]	-	**2.55**
	(F) DDFlow [27]	-	2.97
	(G) UnRAFT (5 M)	2.89	2.90
	(H) UnRAFT-S (1 M)	3.44	3.37

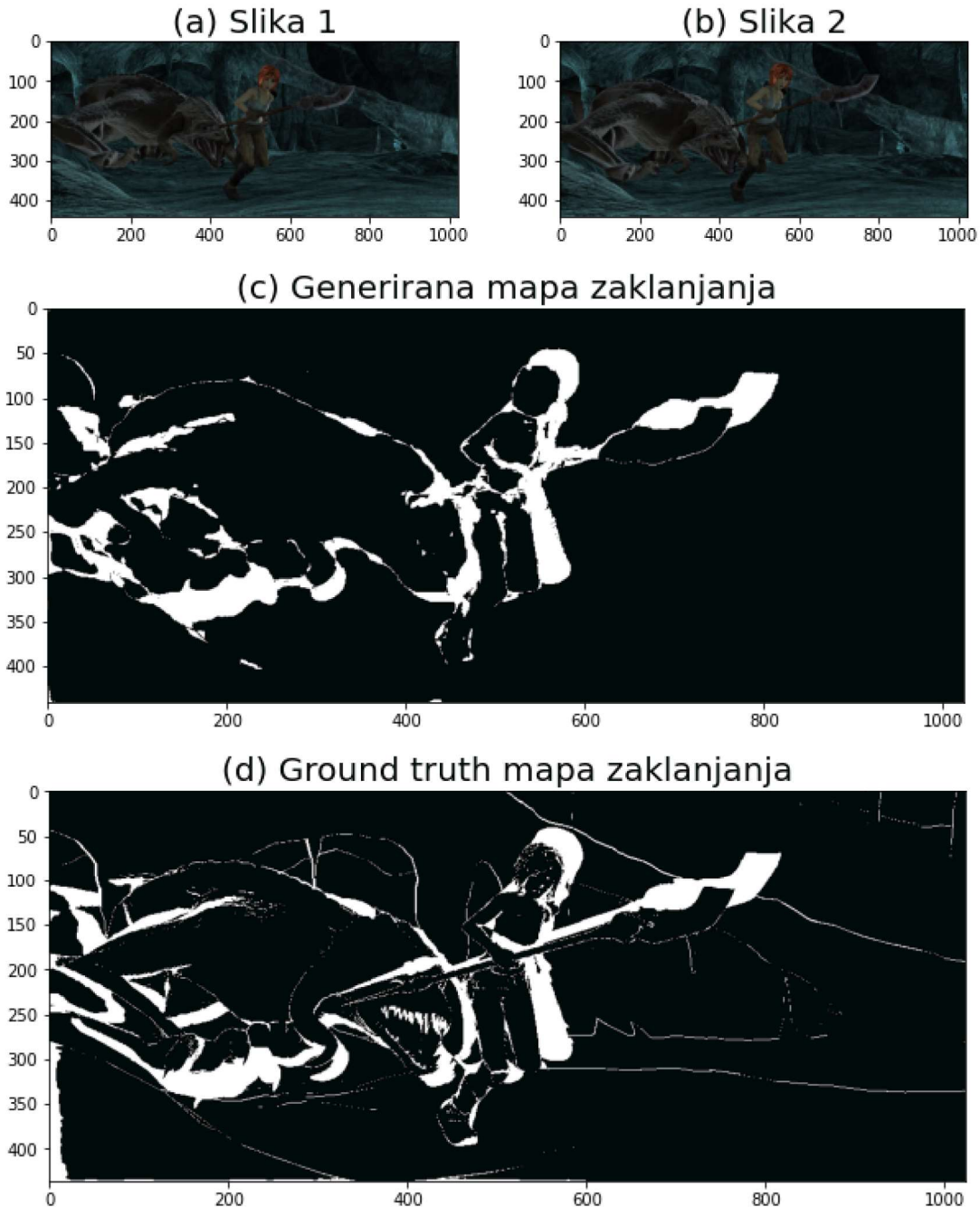
Tablica 7.6: Usporedba rezultata različitih modela postignutih na Flying Chairs test skupu. Modele smo učili samo na skupu Flying Chairs train. Prvi odjeljak tablice prikazuje rezultate nekih nadziranih modela, a drugi odjeljak uspoređuje nenadzirane modele. Naš model koji je učen nenadziranim postupkom opisanim u ovom radu je označen s UnRAFT (G) odnosno UnRAFT-S (H). Podebljani rezultati predstavljaju najbolje modele u odgovarajućoj skupini. Posebno je zanimljivo usporediti rezultate na skupu za učenje i skupu za evaluiranje. Kod našeg samonadziranog modela, gotovo da nema razlike u rezultatima na oba skupa. To možemo objasniti činjenicom da tijekom učenja ne koristimo točne oznake pa se model ne prilagođava specifičnostima skupa za učenje nego uči dobro generalizirati.

7.4. Vizualizacija procjenjenog toka

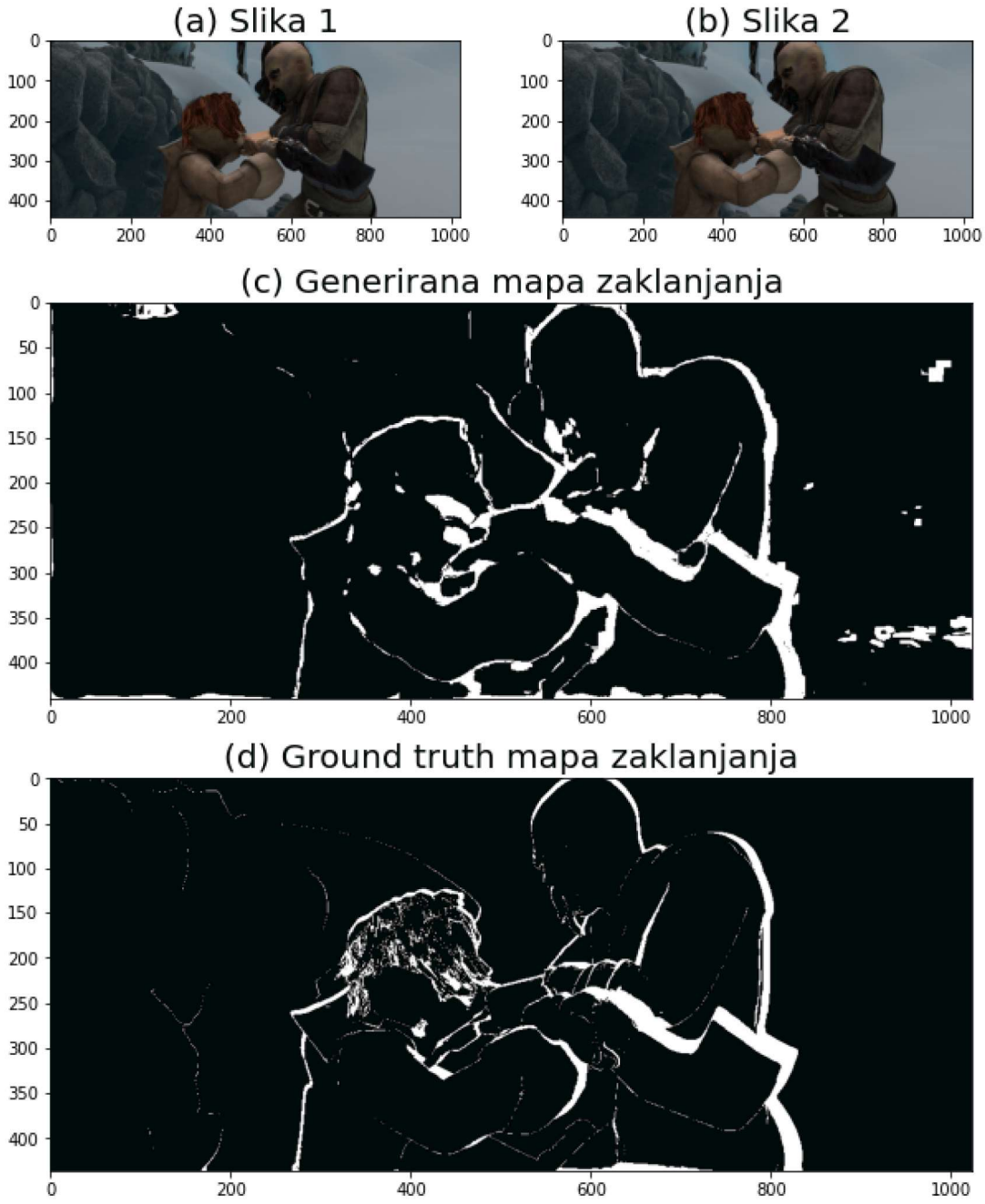
Procjenjeni optički tok možemo vizualizirati preslikavanjem vektorskog polja u HSV sustav boja kako je opisano u odjeljku 2.2. Osim toga, možemo vizualizirati mape zaklanjanja generirane našim nenadzirano naučenim modelom. Sintel podatkovni skup uključuje oznake ispravnih mapa zaklanjanja za svaki par slika iz skupa. Slike 7.8 i 7.9 prikazuju mape zaklanjanja generirane našim nenadziranim modelom i točne mape zaklanjanja koje su pripremili tvorci skupa Sintel.

U nastavku ćemo vizualizirati procijenu optičkih tokova uz vizualnu usporedbu s točnim optičkim tokovima dostupnima u podatkovnim skupovima. Slike 7.10, 7.11 i 7.12 prikazuju procijenjene tokove na nekim parovima iz skupa Sintel clean train koji smo koristili za samonadzirano učenje bez oznaka. Slike 7.14, 7.15 i 7.16 vizualiziraju tok na nekim parovima iz skupa Flying Chairs test generiran modelom učenim samonadzirano na Flying Chairs train (model (b) iz tablice 7.2).

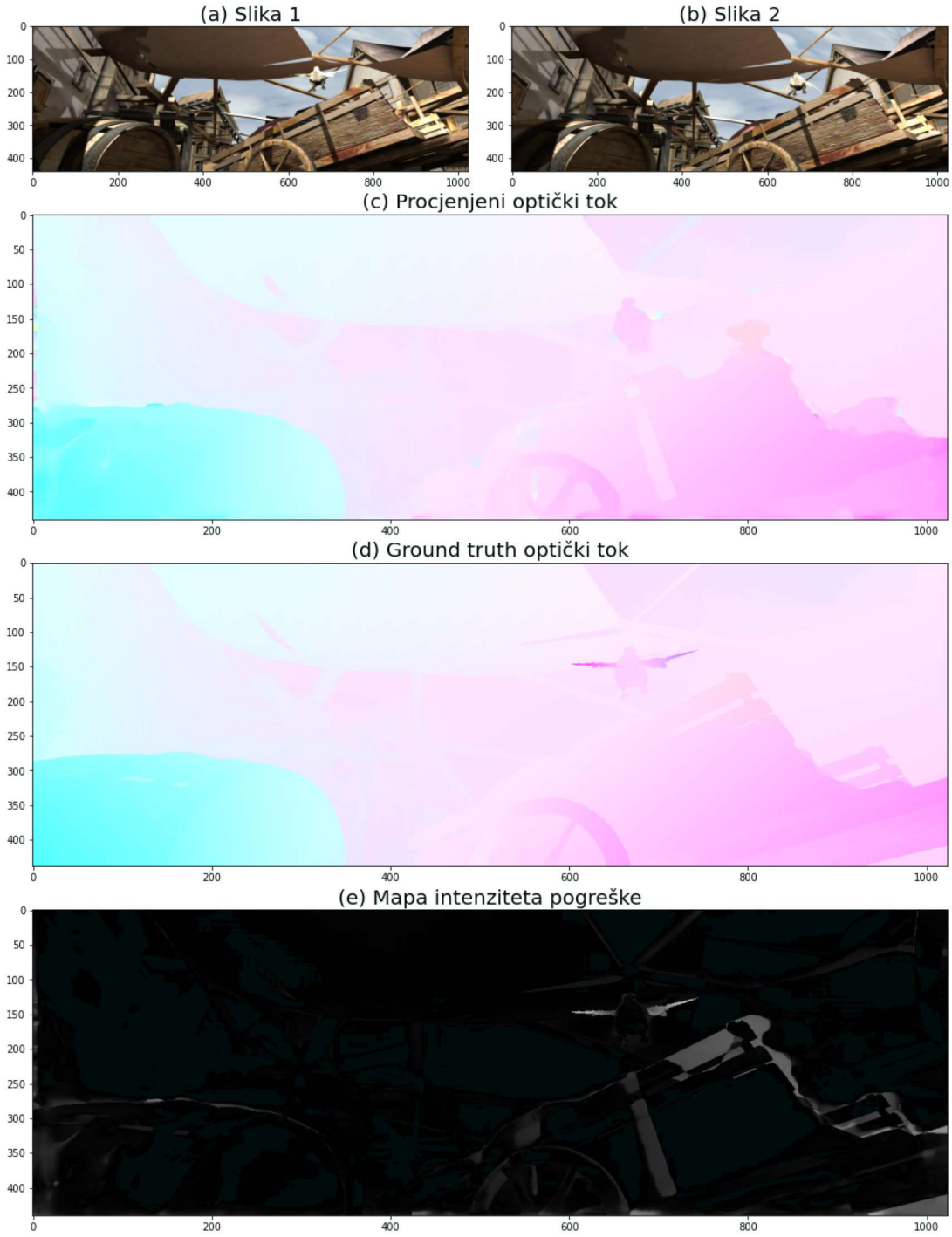
Slike 7.13 i 7.17 prikazuju neuspješne procijene toka u vrlo teškim pikselima. Iako se oba primjera nalaze u skupu za učenje, model nije uspio naučiti ispravna uparivanja. Budući da ne koristimo oznake za učenje, nenadzirani modeli nemaju šanse prepoznati ispravan tok u slučajevima velikih pokreta i zaklanjanja.



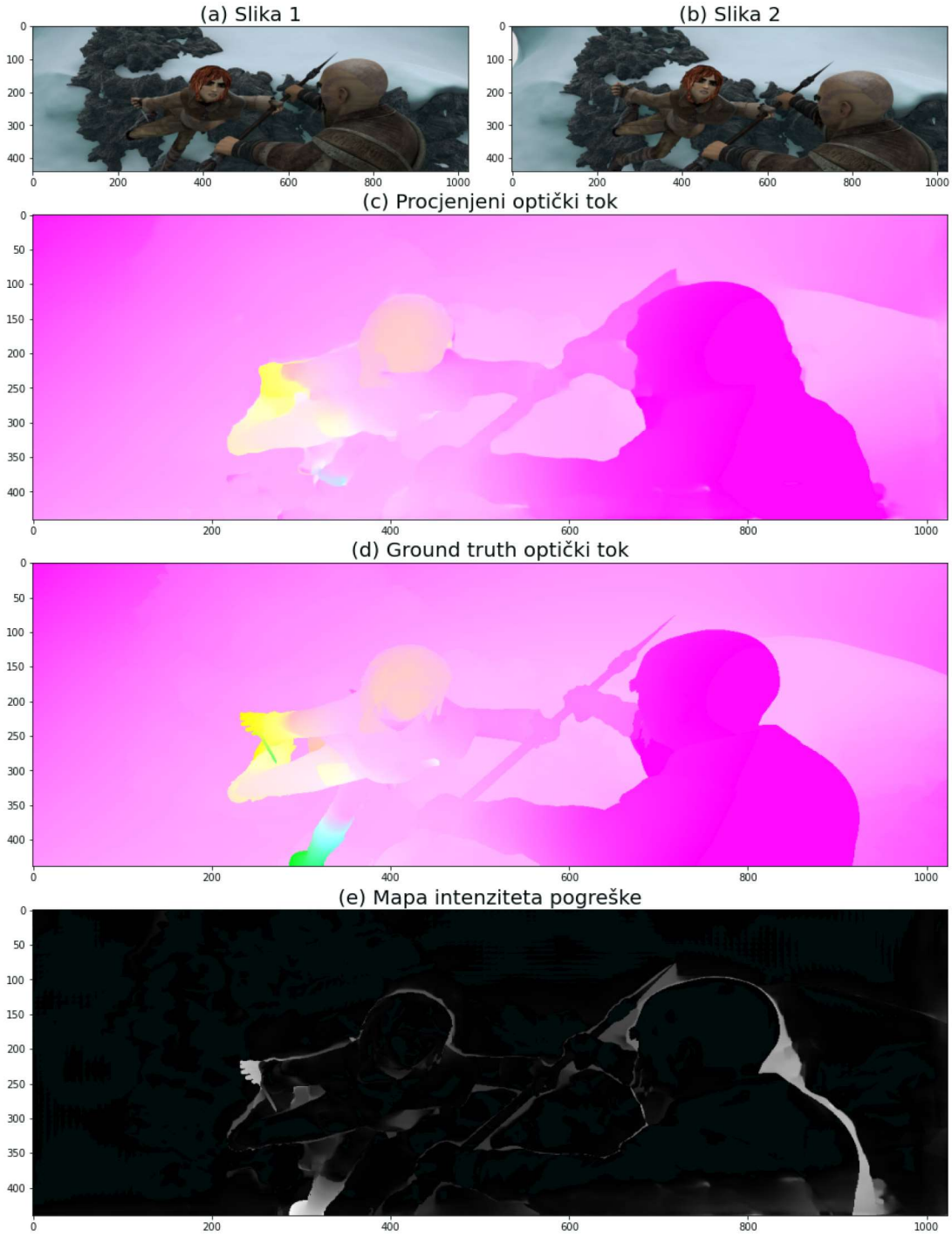
Slika 7.8: Vizualizacija generirane mape zaklanjanja na paru slika iz Sintel clean train skupa te usporedba s točnom mapom zaklananja.



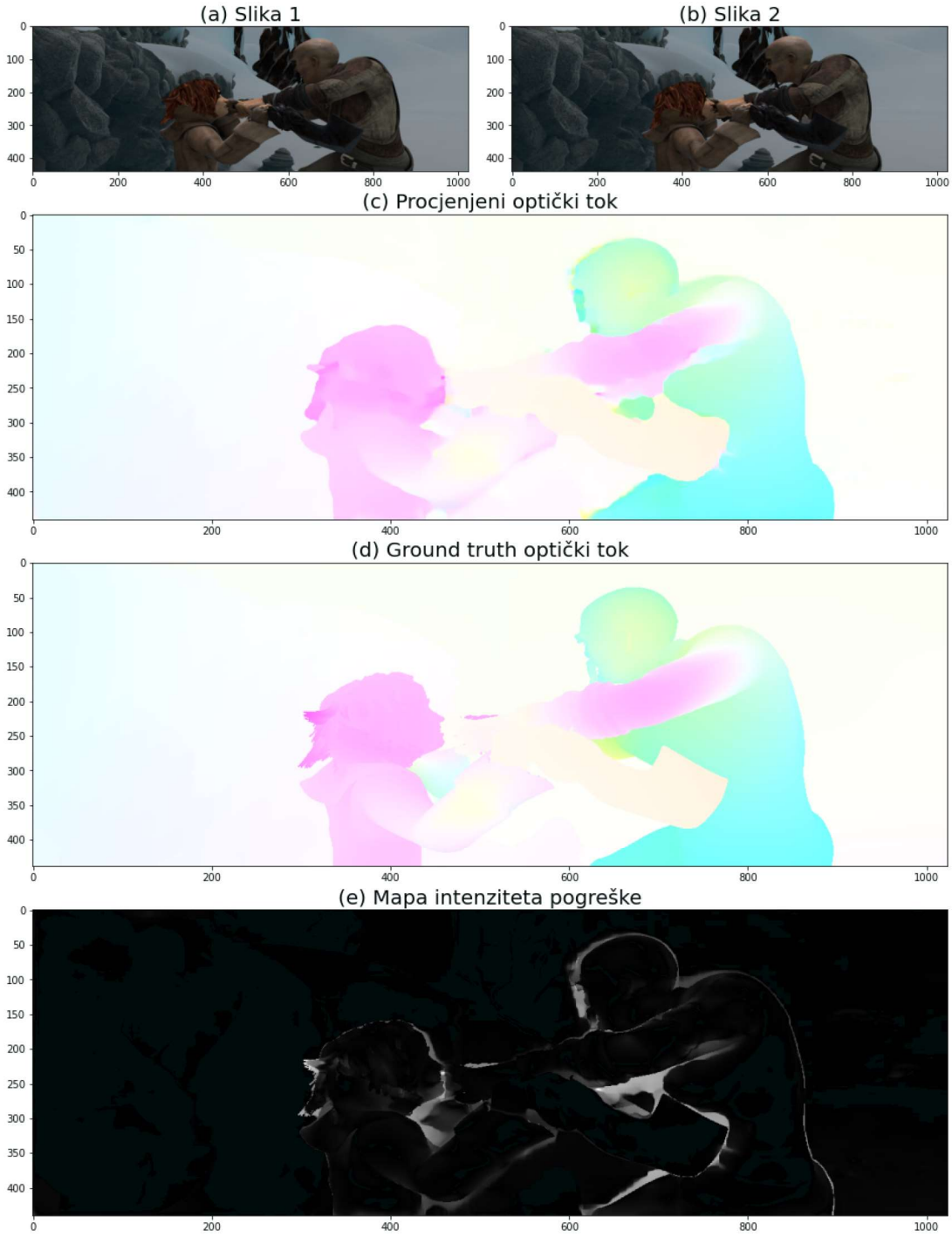
Slika 7.9: Vizualizacija generirane mape zaklanjanja na paru slika iz Sintel clean train skupa te usporedba s točnom mapom zaklananja.



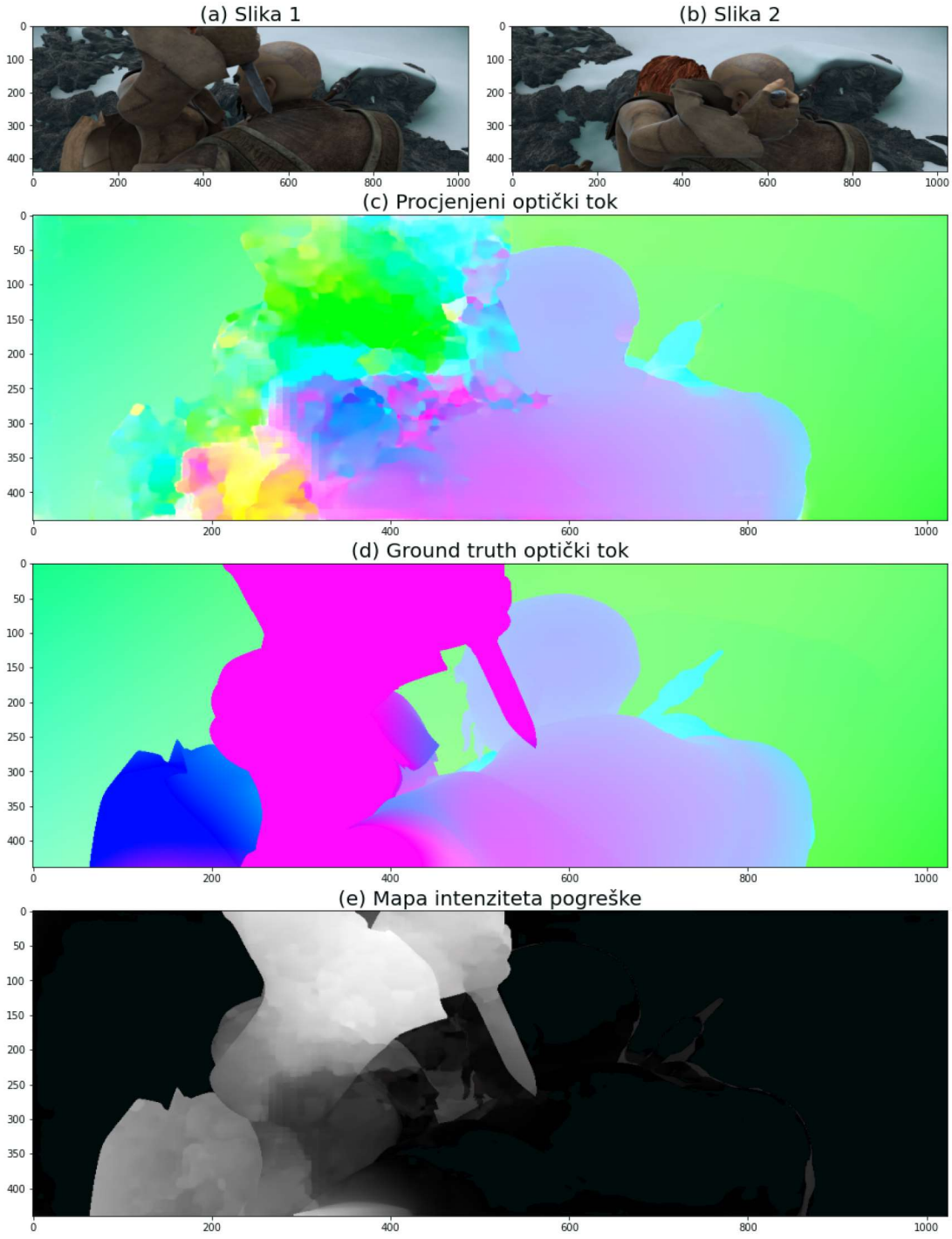
Slika 7.10: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Sintel clean train te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



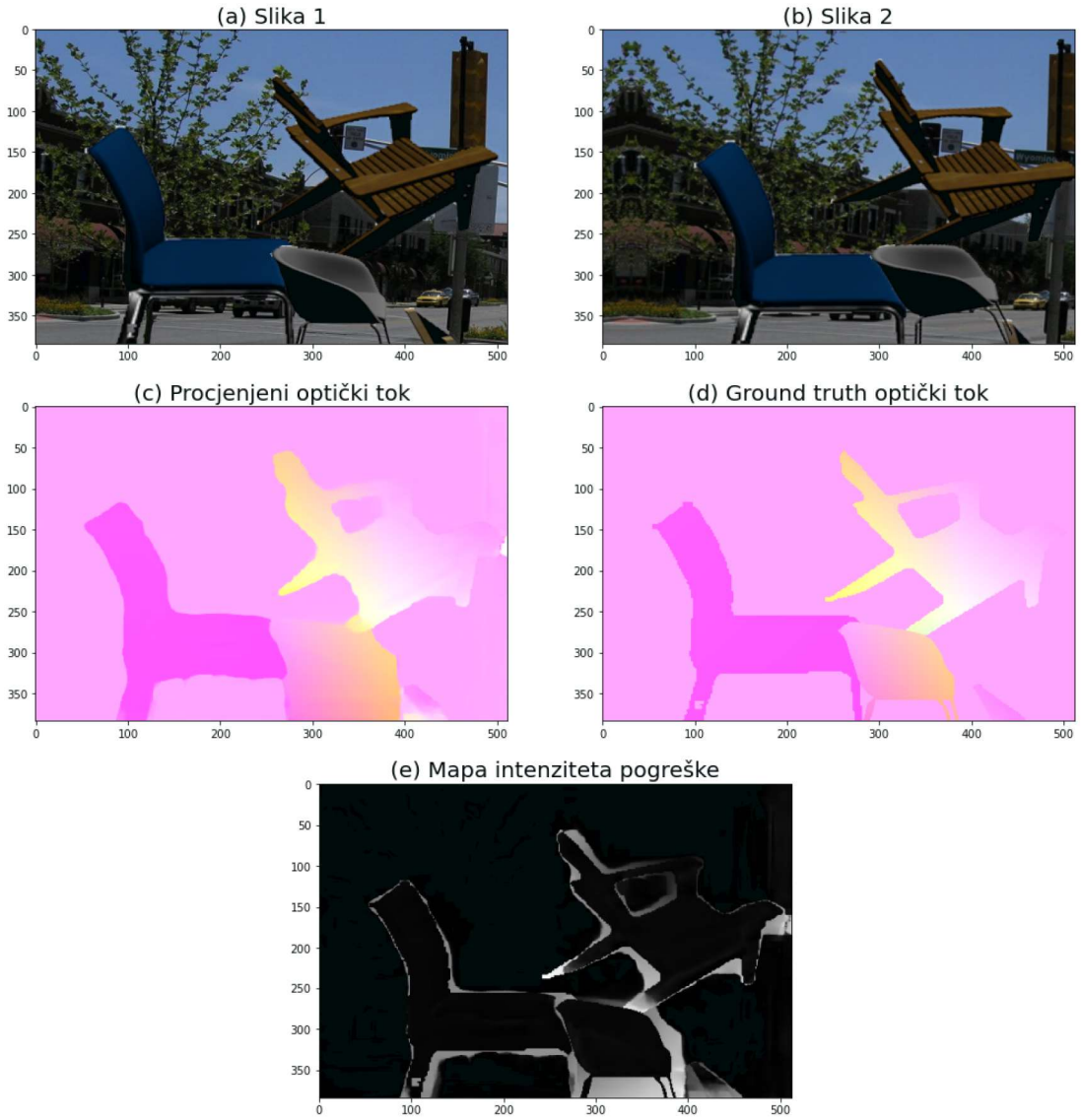
Slika 7.11: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Sintel clean train te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



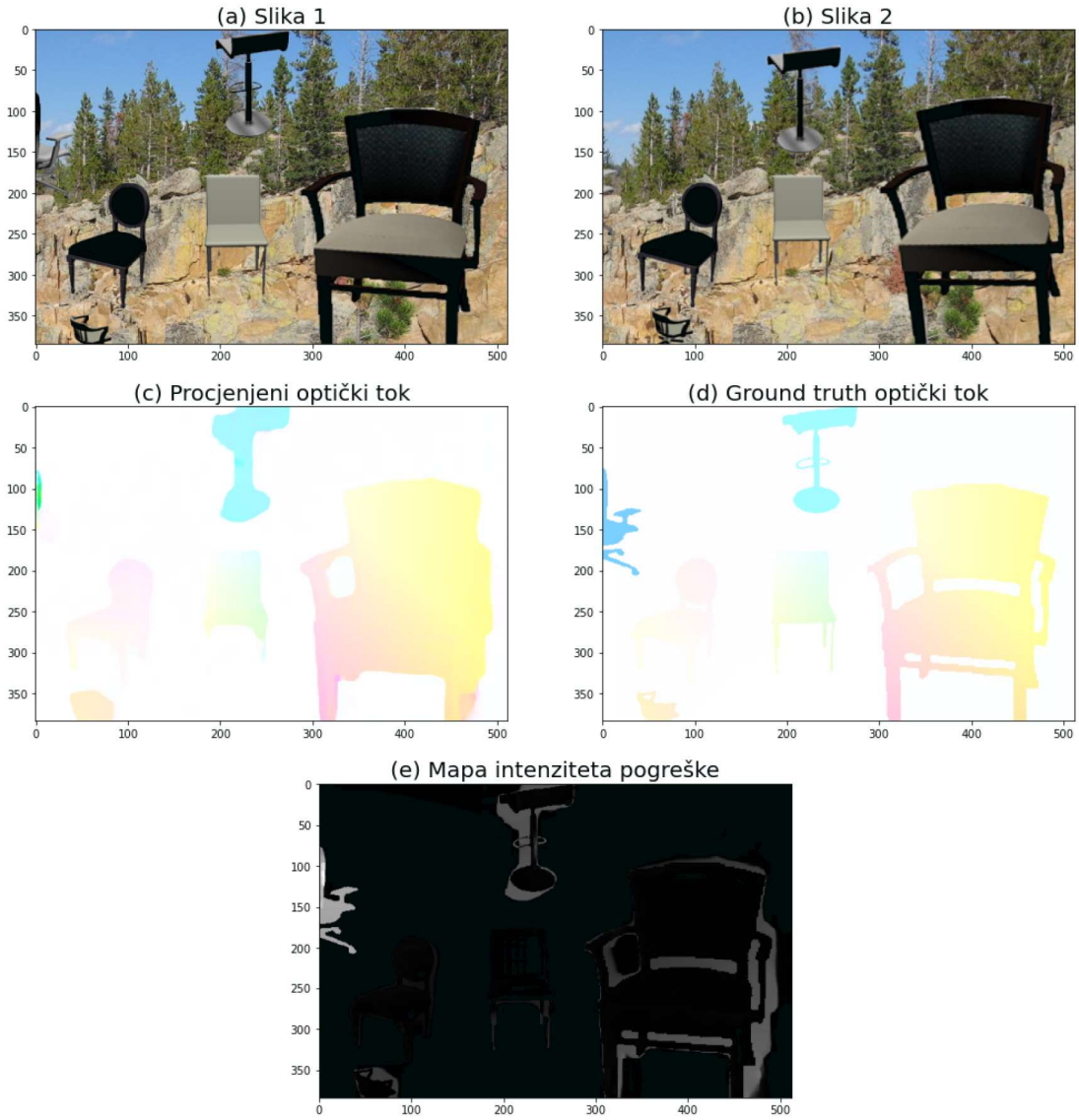
Slika 7.12: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Sintel clean train te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



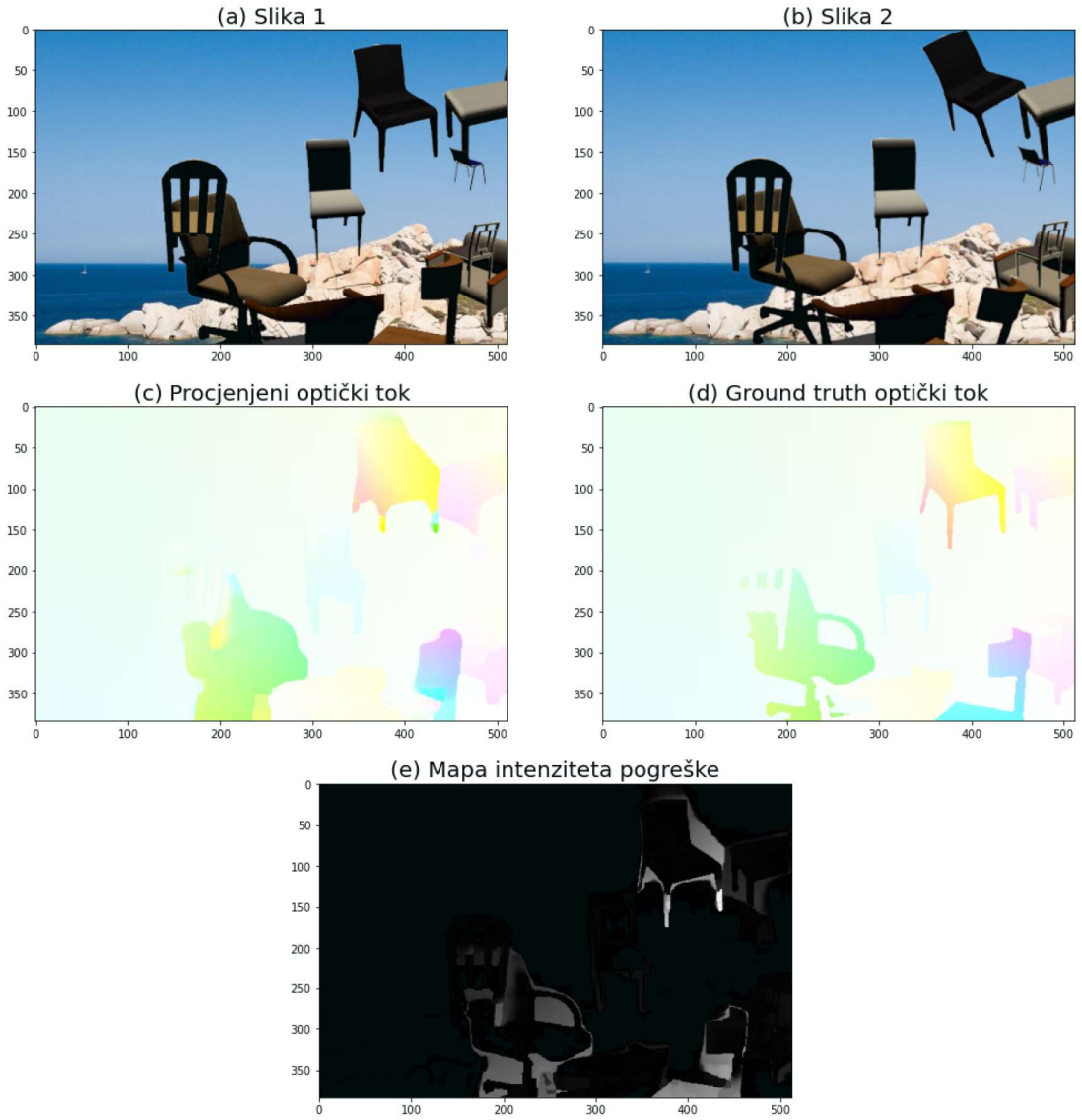
Slika 7.13: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Sintel clean train te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu. Ova slika pokazuje primjer neuspješne procijene toka. Iako je ovaj par slika korišten tijekom samonadziranog učenja, procijena je vrlo teška i model ne uspjeva upariti piksele zbog velikog broja zaklanjanja i velikih pokreta. Samonadzirani modeli poput našeg nemaju velike šanse u ovakvim primjerima zbog toga što tijekom učenja ne koristimo točne oznake, a model ne može sam riješiti ovako težak slučaj.



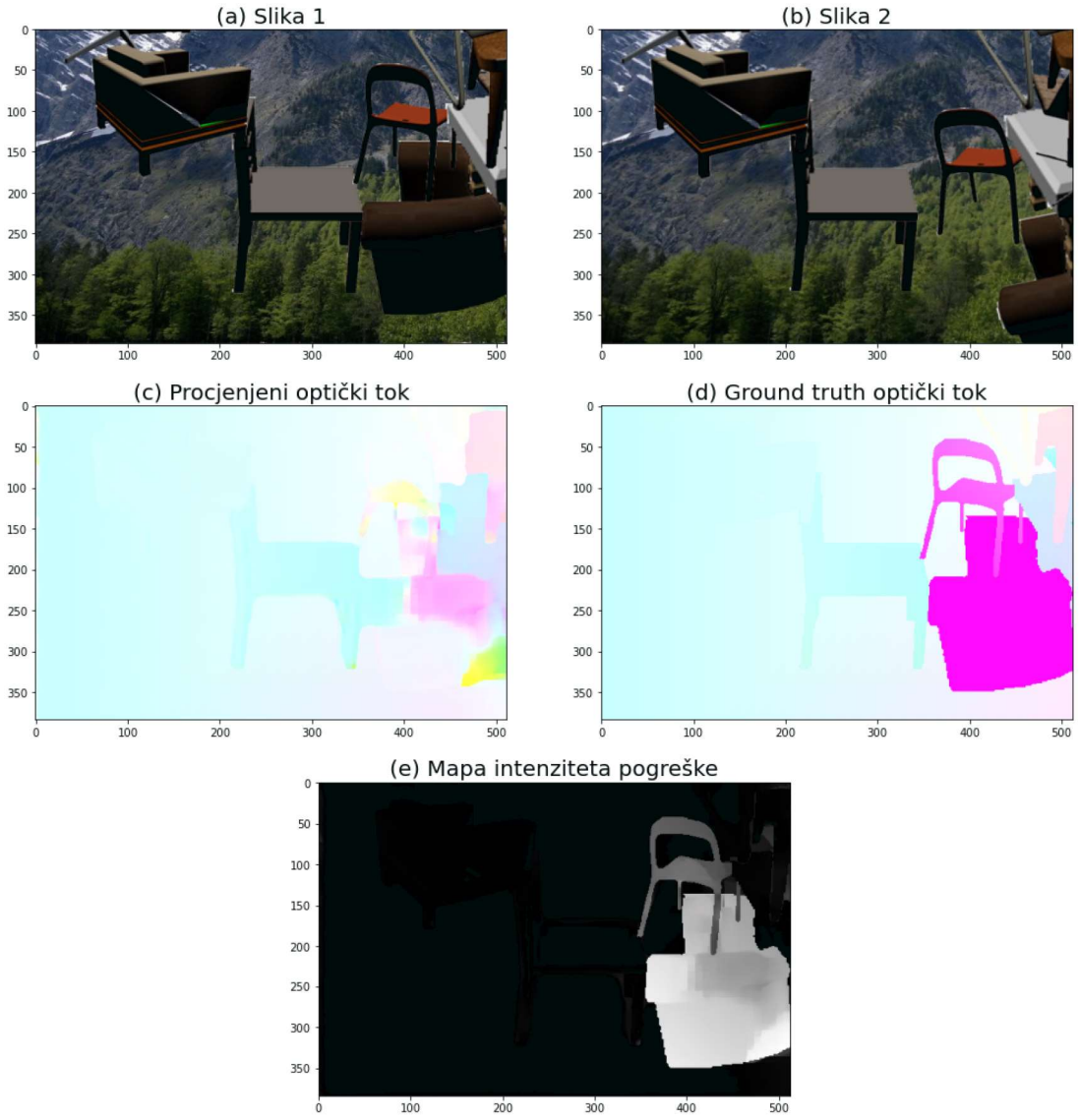
Slika 7.14: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Flying Chairs test te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



Slika 7.15: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Flying Chairs test te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



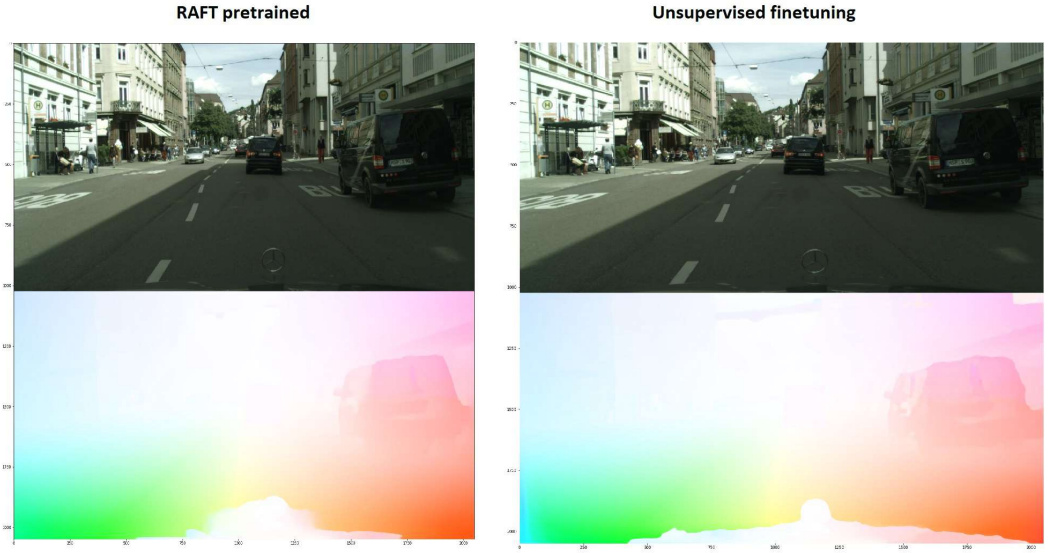
Slika 7.16: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Flying Chairs test te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu.



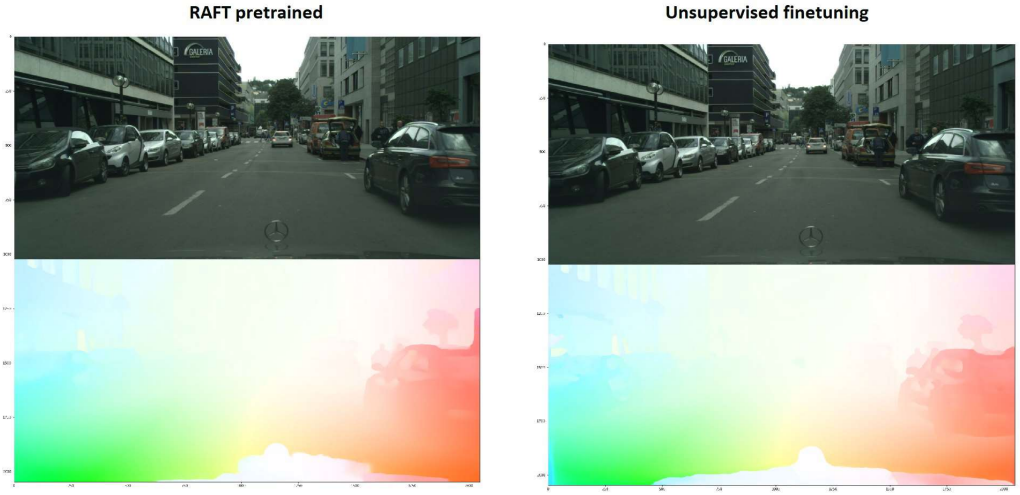
Slika 7.17: Vizualizacija procijenjenog optičkog toka (c) na paru slika (a) i (b) iz skupa Flying Chairs train te usporedba s točnim optičkim tokom (d). Slika (e) pokazuje pogrešku u procjeni toka za svaki pojedini piksel. Veći intenzitet u (e) predstavlja veću pogrešku u odgovarajućem pikselu. Ova slika pokazuje primjer neuspješne procijene toka. Iako je ovaj par slika korišten tijekom samonadziranog učenja, procijena je vrlo teška i model ne uspjeva upariti piksele zbog velikog broja zaklanjanja i velikih pokreta. Samonadzirani modeli poput našeg nemaju velike šanse u ovakvim primjerima zbog toga što tijekom učenja ne koristimo točne oznake, a model ne može sam riješiti ovako težak slučaj.

7.5. Utjecaj ugađanja na Cityscapes podatkovnom skupu

Cityscapes je podatkovni skup koji ne posjeduje točne oznake optičkog toka. Zbog toga nemamo mogućnost ugađanja nadziranih modela kako bi povećali točnost na tom specifičnom skupu i tu primjećujemo veliku prednost nenadziranih modela kao što je naš UnRAFT. Za potrebe korištenja modela na skupu Cityscapes, imamo mogućnost ugađanja modela na skupu jer ionako ne koristimo oznake tijekom faze učenja. Slike 7.18 i 7.19 vizualno uspoređuju procjenu toka nadziranog modela kojeg predlažu autori [43] (lijevo) i našeg samonadziranog modela nakon ugađanja na skupu Cityscapes (desno). Nakon ugađanja, vidljivo je poboljšanje u procijeni toka u području poklopca motora vozila iz kojeg se snima.



Slika 7.18: Vizualna usporedba optičkog toka generiranog nadziranim modelom (lijevo) i našim samonadziranim uz fino ugađanje bez oznaka na skupu Cityscapes (desno).



Slika 7.19: Vizualna usporedba optičkog toka generiranog nadziranim modelom (lijevo) i našim samonadziranim uz fino ugađanje bez oznaka na skupu Cityscapes (desno).

8. Zaključak

U ovom radu opisan je postupak samonadziranog učenja modela RAFT za procjenu optičkog toka. Osnovne značajke modela su iterativan pristup, postepeno ažuriranje trenutne procjene toka i dijeljenje parametara slojeva među iteracijama, a predloženi postupak učenje obavlja u dvije faze. Naša implementacija je vrlo efikasna te uz odgovarajuće računalne resurse model uspješno obrađuje video snimke visoke rezolucije u stvarnom vremenu što model čini prikladnim za korištenje u različitim primjenama. Uz to, razvijeni nenadzirani postupak učenja omogućuje primjenu modela u proizvoljnoj domeni budući da ne zahtijevamo poznavanje oznaka toka tijekom učenja.

Pokazujemo da su rezultati na mjerodavnim skupovima dobiveni predloženim nenadziranim postupkom vrlo usporedivi s najboljim poznatim nenadziranim modelima uz vrlo dobru generalizaciju na skupovima koji nisu viđeni tijekom učenja. Također, pokazujemo utjecaj nenadziranog ugađanja na skupovima koji ne posjeduju točne oznake toka kao što je skup Cityscapes.

Zanimljivi pravci za budući rad uključuju analizu utjecaja različitih postavki učenja kao što su veličina grupe za učenje (engl. batch size) i broj iteracija uz različite stope učenja što ipak zahtjeva značajnu količinu potrebnih računalnih resursa. Budući rad može istražiti i različite formulacije funkcija gubitaka uz korištenje, primjerice, gubitka glatkoće toka (engl. smoothness loss) kao što predlažu autori u [22].

LITERATURA

[1] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE transactions on pattern analysis and machine intelligence*, (4):384–401, 1985.

[2] Antonio Anđelić. Konvolucijski modeli za optički tok. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2020. URL `http://www.zemris.fer.hr/~ssegvic/project/pubs/andelic20ms.pdf`.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, i Geoffrey E. Hinton. Layer normalization, 2016.

[4] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, i Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[5] Patrizia Baraldi, Enrico De Micheli, i Sergio Uras. Motion and depth from optical flow. U *Alvey Vision Conference*, stranice 1–4, 1989.

[6] Steven S. Beauchemin i John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[7] Karla Brkić, Srđan Rašić, Axel Pinz, Siniša Šegvić, i Zoran Kalafatić. Combining spatio-temporal appearance descriptors and optical flow for human action recognition in video data. *arXiv preprint arXiv:1310.0308*, 2013.

[8] Thomas Brox, Christoph Bregler, i Jitendra Malik. Large displacement optical flow. U *2009 IEEE Conference on Computer Vision and Pattern Recognition*, stranice 41–48. IEEE, 2009.

[9] Qifeng Chen i Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 4706–4714, 2016.

[10] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, i Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, EMNLP 2014. URL `https://arxiv.org/abs/1406.1078v3`.

[11] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, i Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[12] Denis Fortun, Patrick Bouthemy, i Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

[13] Zoubin Ghahramani. Unsupervised learning. U *Summer School on Machine Learning*, stranice 72–112. Springer, 2003.

[14] David Hafner, Oliver Demetz, i Joachim Weickert. Why is the census transform good for robust optic flow computation? U *International Conference on Scale Space and Variational Methods in Computer Vision*, stranice 210–221. Springer, 2013.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition, 2015.

[16] Berthold KP Horn i Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[17] Han Hu, Chongtai Chen, Bo Wu, Xiaoxia Yang, Qing Zhu, Yulin Ding, et al. Texture-aware dense image matching using ternary census transform. 2016.

[18] Tak-Wai Hui, Xiaoou Tang, i Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 8981–8989, 2018.

[19] Junhwa Hur i Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 5754–5763, 2019.

[20] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, i Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 2462–2470, 2017.

[21] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, i Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. U *Proceedings of the European Conference on Computer Vision (ECCV)*, stranice 690–706, 2018.

[22] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, i Anelia Angelova. What matters in unsupervised optical flow. *arXiv preprint arXiv:2006.04902*, 1(2):3, 2020.

[23] Clemens Kirisits José A. Iglesias. Convective regularization for optical flow. *Variational Methods*, stranica 184–201, Dec 2016. doi: 10.1515/9783110430394-005. URL `http://dx.doi.org/10.1515/9783110430394-005`.

[24] Simeon Kostadinov. Understanding gru networks, 2017. URL `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`.

[25] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, i Tom Goldstein. Visualizing the loss landscape of neural nets, 2018.

[26] Gal Lifshitz i Dan Raviv. Unsupervised optical flow using cost function unrolling. *arXiv preprint arXiv:2011.14814*, 2020.

[27] Pengpeng Liu, Irwin King, Michael R Lyu, i Jia Xu. Ddflow: Learning optical flow with unlabeled data distillation. U *Proceedings of the AAAI Conference on Artificial Intelligence*, svezak 33, stranice 8770–8777, 2019.

[28] Pengpeng Liu, Michael Lyu, Irwin King, i Jia Xu. Selflow: Self-supervised learning of optical flow. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 4571–4580, 2019.

[29] Ilya Loshchilov i Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[30] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, i T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. U *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. URL `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`. arXiv:1512.02134.

[31] Simon Meister, Junhwa Hur, i Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. U *Proceedings of the AAAI Conference on Artificial Intelligence*, svezak 32, 2018.

[32] Moritz Menze, Christian Heipke, i Andreas Geiger. Discrete optimization for optical flow. U *German Conference on Pattern Recognition*, stranice 16–28. Springer, 2015.

[33] Shahriar Negahdaripour. Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):961–979, 1998.

[34] Javier Sánchez Pérez, Enric Meinhardt-Llopis, i Gabriele Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013.

[35] Anurag Ranjan i Michael J Black. Optical flow estimation using a spatial pyramid network. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 4161–4170, 2017.

[36] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, i Aleksander Madry. How does batch normalization help optimization?, 2019.

[37] Josip Saric, Marin Orsic, Tonci Antunovic, Sacha Vrazic, i Sinisa Segvic. Warp to the future: Joint forecasting of features and feature motion. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 10648–10657, 2020.

[38] Siddharth Sharma, Simone Sharma, i Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 05 2020. doi: 10.33564/IJEAST.2020.v04i12.054.

[39] Igor Smolkovič. Analiza i vrednovanje odabranih izvedbenih detalja postupaka optičkog toka. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike

i računarstva, 2014. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/smolkovic14ms.pdf.

[40] Deqing Sun, Stefan Roth, JP Lewis, i Michael J Black. Learning optical flow. U *European Conference on Computer Vision*, stranice 83–97. Springer, 2008.

[41] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, i Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 8934–8943, 2018.

[42] Zachary Teed i Xiangyu Chen. Raft. 2020. URL https://github.com/princeton-vl/RAFT.

[43] Zachary Teed i Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[44] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, i Wei Xu. Occlusion aware unsupervised learning of optical flow. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, stranice 4884–4893, 2018.

[45] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, stranice 6278–6287, 2020.

[46] Yi-Tong Zhou i Rama Chellappa. Computation of optical flow using a neural network. U *ICNN*, stranice 71–78, 1988.

Nenadzirano učenje optičkog toka

Sažetak

Zadatak procijene optičkog toka jedan je od neriješenih problema u području računalnog vida. Optički tok ima široku primjenu u praksi te se zbog toga pojavljuju brojni pokušaji rješavanja ovog problema. Počevši od formulacije problema kao optimizacijskog postupka te korištenja ručno dizajniranih opisnika, današnje stanje tehnike u području optičkog toka postignuto je dubokim konvolucijskim modelima. Primjer takvog dubokog konvolucijskog modela je RAFT koji se sastoji od tri glavne komponente: kodera značajki, korelacijskog sloja i modula za ažuriranje. Osnovno svojstvo RAFT-a je iterativan pristup i korištenje volumena cijene sa svim parovima piksela. U ovom radu predlažemo kompletan postupak samonadziranog učenja modela RAFT. Učenje obavljamo u dvije faze tako da procijenu toka u nezaklonjenim pikselima iz prve faze koristimo kao točne oznake za vođenje učenja u umjetno zaklonjenim dijelovima tijekom druge faze. Najvažnija karakteristika predloženog postupka je mogućnost učenja na parovima slika bez poznavanja točnog toka čime uklanjamo ograničenja na domenu primjene. Procijenu evaluiramo na skupovima Sintel i Flying Chairs te pokazujemo da naš model postiže rezultate usporedive s najboljim poznatim modelima, a navodimo i mogućnosti budućeg napretka.

Ključne riječi: računalni vid, optički tok, duboko učenje, CNN, PyTorch, RAFT, samonadzirano učenje, nenadzirano učenje, analiza kretanja slike, raspoznavanje uzoraka, računalna znanost

Unsupervised learning of optical flow

Abstract

The task of estimating optical flow is one of the unsolved problems in the field of computer vision. Optical flow has a wide application in practice and therefore there are numerous attempts to solve this problem. Starting from the formulation of the optical flow as an optimization problem and the use of hand-crafted descriptors, the current state-of-the-art in the field of optical flow has been achieved by deep convolutional models. An example of such a deep convolutional model is RAFT which consists of three main components: a feature encoder, a correlation layer and an update operator. The main feature of RAFT is the iterative approach and the use of cost volume with all pairs of pixels. In this paper, we propose a method for self-supervised learning of the RAFT model. We perform the learning in two phases by using the flow estimation in the non-occluded pixels from the first phase as ground-truth to guide learning in the manually occluded areas during the second phase. The proposed method is able to learn on pairs of images without knowing the ground-truth flow, thus removing restrictions on the field of application. We evaluate the estimations at the Sintel and Flying Chairs and show that our model achieves results comparable to the best known models, and we also discuss the possibilities for the future work.

Keywords: computer vision, optical flow, deep learning, CNN, PyTorch, RAFT, self-supervised learning, unsupervised learning, image motion analysis, pattern recognition, computer science